Development of a platform for the copyright protection of multimedia content

author

November 27, 2012

Abstract

This paper presents a software platform that consists of a combination of watermarking and fingerprinting techniques to help protect authorship and copyright of multimedia content. The software that has been developed, provides content distributors and authors with a trusted system that allows the former to develop new business models while preserving the authorship rights of the latter. The proposed system offers the distribution of multimedia content via a Web platform, providing mechanisms for tracing dishonest users that illegally redistribute their content.

1 Introduction

Distribution and playback of digital content (image, audio, video, text, ...) using a personal computer has become a trivial matter. This ease of use leads to problems concerning the protection of copyright and distribution rights. The scientific community has been devoting extraordinary efforts to devise a copyright protection system that helps to prevent these illegal practices. Of course, data encryption provides protection up to the transport layer, but when an authorized dishonest user decrypts his/her content, nothing can prevent him/her from distributing the content without any concern to be indicted or punished for his/her actions. Watermarking schemes appear to be a possible solution to this problem. These schemes allow to embed information about the owner (brand) in a given original content. Unfortunately, watermarking alone does not protect against illegal re-distribution. However, when combined with digital fingerprinting techniques [1], both mechanisms offer a good solution to deter potential fraudulent users from re-distributing illegal content. To make distribution systems work properly, the solution must be robust to attacks such as image processing operations, lossy compression, geometric transformation, combination with additive noise, and/or collusion attacks.

The work proposed in this article consists on the design and implementation of an empiric and portable platform that satisfies the following objectives:

- Provide a public and secure platform capable of tracing users that perform illegal digital video re-distribution.
- Check on a practical level the robustness of watermarking and fingerprinting algorithms working together.

The platform, available at `http://drm.upc.edu`, is presented as a content manager system (CMS), liaising between authors and customers. The aim is to provide a trusted environment that offers protection against illegal re-distribution of authorized content. The protection provided is achieved by means of combining watermarking and fingerprinting techniques for the case of MPEG-2 digital content. The generated watermarks, as well as the insertion algorithm have been designed in such a way that guarantees robustness against common attacks, while avoiding content degradation.

The paper is structured as follows; Section 2 describes the working scenario, paying special attention to the platform functional and architectural key points, such as security, modularity and flexibility. Section 3 discusses watermarking mechanisms, as well as the implementation internals of the watermarking layer. Section 4 describes the different entities and their interrelationship. Finally, conclusions are given in Section 5

2 Working Scenario

Consider an scenario where a digital content provider offers its users copies of digital video for their amusement. Of course, the provider desires as much protection as possible against illegal re-distribution of these files to third parties. Authorized users access the provider's Web portal to download their favorite movies. Once the user has purchased a given content he is free to illegally distribute it via P2P platforms or upload it to content distribution platforms such as Youtube or Fileserve, among others. The content provider, after detecting that one of its products is freely available on the Internet realizes of a security leak in his business chain, but there is not much more that he can do. It would be nice for him/her to at least identify the user that purchased the copy that has been illegally re-distributed. This user may have incurred in a contract fault by distributing content without specific authorization, allowing the company to take internal actions over the user's account, regardless of the legal actions that could be taken over the physical person.

The work we present aims at filling the gap between detection of a dishonest distribution and the actions taken on the accused individual. This may be performed by identifying the owner of the illegally redistributed copy.

So, in order to be able to identify the owner of a given a video file, there must be an association between the user and the copy. This can be achieved by means of inserting some information inside the content that may identify the user once the content is recovered, this embedded information is called a mark. Of course there are some requirements [2] that this mark has to fulfill:

- **Perceptually invisible**: perceptual invisibility is a wanted characteristic of the mark. The process of embedding the information into the data is really important. In one hand, the introduced mark cannot be perceptually noticed by viewers. On the other hand, once introduced it cannot be easily removed, at least without much degradation of the original data.

- **Statistically invisible**: each mark must be statistically uncorrelated to any other mark or the content itself.

- **Robustness**: The mark must be robust to modifications as, for example, image processing.
- **Unambiguous**: A mark should unambiguously identify the authorized owner of the content in which has been introduced.
- **Complexity**: The complexity of a mark is proportional to its degree of efficiency. A more complex mark is also more difficult to remove.
- **Low error bit rate**: When recovering the mark it has to be guaranteed that it is "impossible" to obtain a mark that belongs to a user different than the original one. "Impossible" here means with very low probability.

Clearly there is a need to embed customer information into delivered multimedia products to ensure copyright protection. The presented platform pays special attention to this concern and proposes the usage of user and server Public-Key certificates to protect user-to-platform and platform-to-platform transactions.

3 Implementation Details

The aim of this platform is to guarantee the correct management of digital rights (both copyright and distribution rights). This goal is accomplished by means of watermarking algorithms and fingerprinting codes. In this section, the implementation of these mechanisms is discussed. On one hand, the watermarking layer discusses how the marks are embedded into the video content. On the other hand, fingerprinting algorithms generate the embedded marks.

3.1 Watermarking Layer

Watermarking is a technique for embedding information into data files. When using watermarking schemes, all the copies of a file contain a mark, that can be recovered at any time.

From the detection point of view watermarking systems can be classified in two groups: blind watermarking and non-blind watermarking. The first one assumes that the original content is not needed to perform the recovery of the mark. The second algorithm assumes that the original content can be used by the decoder in order to improve the performance of the whole system. Since in our working scenario, it can be assumed that the decoder has access to the original content, we have decided to use non-blind watermarking. An extensive analysis about the drawbacks of blind watermarking systems is given in [3].

Usually watermarking is performed in the frequency domain. There are different domain transforms we can use, for example, the Fast Fourier transform (FFT) or the Discrete Cosine Transform (DCT). In [3] has been shown that DCT gives a better answer to image watermarking needs. After the transformation into the frequency domain a mark is added to the signal as it is explained in Appendix A. The inverse process transforms again the signal to the spatial domain.

There are many advantages of placing the watermark in the frequency domain. As it also occurs in the spatial domain, different frequency values can be modified. Each frequency coefficient has a perceptual capacity. This perceptual capacity represents the capacity of each coefficient to receive additional information (watermark), without introducing perceptual differences in the content. Note that the Human Visual System model (HVS) naturally works with frequencies. So working in the frequency domain helps adjusting the addition of information together with the HVS sensitivity.

3.2 Fingerprinting Layer

Roughly speaking, the fingerprinting layer will generate a different mark for each distributed copy of a multimedia content. This mark will be embedded by the watermarking layer into a verbatim copy of the original content in order to generate a new copy for a particular user. When one of these copies is illegally re-distributed, this mark is extracted, and by means of a tracing algorithm, the traitor (the fraudulent user) is identified.

This first approximation is more or less achievable but the real fingerprinting problem consists in finding, for each copy of the original content, the right set of marks that prevent collusion attacks. A collusion attack is performed by a set of users that have acquired different copies of the same multimedia content. Note that, in order to identify every user, every single mark embedded into each copy has to be different. Since the attackers decide to cooperate, they have access to different copies of the same content so they can generate a pirate copy of this content by mixing their copies. In this scenario, the pirate copy will have parts of each original mark but it will be different from any of them. So the aim of a fingerprinting code is find the set of attackers that have taken part in a collusion attack.

Codes that are robust to collusion attacks are called collusion secure codes. The construction of collusion secure codes was first addressed in [1]. In that paper, Boneh and Shaw obtain ($c > 1$)-secure codes, which are capable of identifying a guilty user in a coalition of at most c users with a probability ϵ of failing to do so. The algorithm composes an inner binary code with an outer random code. Therefore, the identification algorithm involves the decoding of a random code, that is known to be a NP-hard problem [4]. Moreover, the length of the code is considerably large for small error probabilities and a large number of users.

To reduce decoding complexity, Barg, Blakley and Kabatiansky in [4] used algebraic-geometric codes together with separating codes to construct fingerprinting schemes. In this way, their system reduces the decoding complexity to $O(poly(n))$ for a code with length n. In [5], Fernandez and Soriano constructed a 2-secure fingerprinting code by concatenating an inner $(2, 2)$-separating codes with an outer IPP code (a code with the Identifiable Parent Property), and also with decoding complexity $O(poly(n))$. In [6], Tardos presents a code which has a length of $O(c^2 log(n/\epsilon))$ and is ϵ-secure against c pirates over n users. These codes represented an important improvement compared to the Boneh and Shaw proposal.

In our implementation, the code and algorithms presented in [5] have been used as fingerprinting codes mainly because in our scenario there is a small number of users

and the size of the attacking coalitions is assumed to be small also. In Appendix A we discuss briefly the construction of these codes.

3.3 Implementation Details of Digital Rights Protection

The watermarking process requires a considerable amount of CPU running time and computer memory, so it is important to use an efficient encoding/decoding process to add the mark to the video. This was the most important factor when deciding how to implement a functional marking software. Implementing an MPEG2 encoder/decoder is difficult and would require a lot of knowledge to make it efficient. To solve this problem we decided to implement the marking algorithm on an already working MPEG2 encoder/decoder, FFmpeg.

FFmpeg was chosen because it is licensed under the GPL, and because it is highly efficient. The work presented in this paper will be mainly focused on adding a new set of filters that allow the embedding and recovery of a mark inside a video stream.

Libavfilter is an FFmpeg library, designed to simplify the creation of filters that can be applied to video processing, regardless of the format of the input and output videos. This is accomplished by decoding the input stream and passing it to the filter a single frame at a time, then the processed frame is coded and saved in the output stream, this flow is represented in Figure 1.

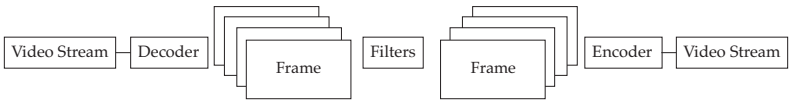


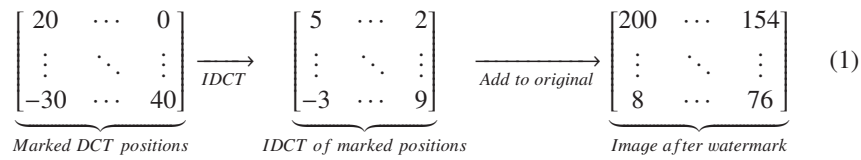
Figure 1: Libavfilter working flow

This flow allows developers of filters to focus on the processing of the image, without handling any of the decoding, encoding or muxing. This is specially interesting when the filter has to be applied in the spatial domain, but in our case the filter has to be applied in the DCT domain. This might seem like a problem, since AVFilter provides us with an image divided in 4 planes. The more usual procedure for modifying the frame would be to perform the DCT of the pixels, modify the desired coefficients and perform the IDCT to convert the image back to the spatial domain. This however can be simplified with the following DCT property.

Definition 1 ***(from [7])*** *Given the integrable functions $f(x)$, $g(x)$ and $h(x)$ we denote their discrete cosine transforms by $\hat{f}(\xi)$, $\hat{g}(\xi)$ and $\hat{h}(\xi)$ respectively:*

For any complex numbers a and b, if $h(x) = a \cdot f(x) + b \cdot g(x)$, then $\hat{h}(\xi) = a \cdot \hat{f}(\xi) + b \cdot \hat{g}(\xi)$.

Based on Definition 1, the embedding process is less costly, since we no longer need to perform the DCT of the original image, instead we can create and empty matrix, fill the desired positions with the marking coefficients, perform the IDCT of the matrix and add it to the original image as seen in (1).



(1)

Once the watermarking and recovery filters were implemented, some tests were conducted in order to prove the robustness of the system. A sample movie with the following properties was used to conduct these tests:

Movie sample	
Duration	101s
Size	47,6MB
Bitrate	4000kb/s
Framerate	23.98fps
Dimensions	848x480
GOP Size [1]	12 frames

This video sample corresponds to a movie trailer freely available on the Internet. All tests where performed trying to simulate an scenario as close to real life as possible. Therefore, the sample was re-encoded several times. For instance, in the scale tests, the movie was first watermarked, then scaled to the desired size. Then it was scaled back to the original size and finally it was processed in order to recover the embedded mark (the movie was decoded and encoded 3 times). The tests were performed this way to simulate the behavior of a dishonest user, who might try to scale the movie in order to remove the embedded mark. Once the movie is recovered, it has to be scaled back and then compared with the original one in order to extract the mark.

The table with the result of the simulations contains the Bit Error Ratio (BER) of each recovered watermark. This rate is the number of erroneous bits divided by the total number of bits that a mark contains.

As we can see in the results table, we obtain a very low BER on most tests, and the only way to introduce a significant degradation to the mark also has a very high impact on the image quality, rendering the sample unwatchable (a blur filter with a factor higher than 7 or a Gaussian filter with a factor higher than 12). Hence we can assert that the implementation of the watermarking algorithm is robust, portable and efficient.

4 Entities and Collaboration

In order to achieve the enumerated requirements, the proposed platform has been divided into the following entities:

[1] Distance between consecutive I-Frames

[2] The formal definition of the mark strength, also called α, is addressed Appendix A

Table 1: Tests results for movie sample

	Mark strength [2]								
	40	35	30	25	20	15	10	5	1
Mark and recover (no filter)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01
Gaussian filter factor 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2
Gaussian filter factor 5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.32
Gaussian filter factor 7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4
Gaussian filter factor 9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.41
Gaussian filter factor 11	0.0	0.0	0.0	0.0	0.0	0.01	0.02	0.1	0.44
Gaussian filter factor 13	0.0	0.0	0.0	0.0	0.004	0.02	0.12	0.23	0.48
Gaussian filter factor 15	0.0	0.006	0.01	0.02	0.07	0.11	0.17	0.3	0.53
Blur filter factor 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3
Blur filter factor 5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.07	0.45
Blur filter factor 7	0.28	0.23	0.26	0.18	0.16	0.24	0.37	0.42	0.49
Scale to 706x400 (5/6)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.21
Scale to 636x360 (3/4)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.32
Scale to 424x240 (1/2)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.37
Scale to 282x160 (1/3)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.51

- **User**: consists on the physical person that interacts with the system via the Web interface. Later, the different roles a user can take will be introduced.
- **File server (FS)**: stores the original files as well as the watermarked copies.
- **Marker server (MS)**: provides marked copies of original files.
- **Tracker server (TS)**: manages the business work flow.
- **Web server (WS)**: provides a Web interface to all platform functionalities.
- **Certification authority (CA)**: manages the certificates that will be used in the communication process, and has real-time certificate revocation status information.

The interaction between these entities provide the end-users with a protected copy of a digital video file. The flow of the platform is the following: *verification, protection* and *delivery*.

During the *verification* phase, both client and platform, must validate each other's credentials in order to allow further transactions to take place.

Considering the nominal usage of the copyright protection platform, someone aiming to request digital contents, the user, is required to own a private certificate (X.509) delivered by the internal certification authority (CA). This authority provides valid certificates to both users and services. This way, all communications carried out with and inside the platform are secured using the provided certificates.

When contacting the platform public URL through a common Web browser, the client will be requested to specify the certificate to be used in the negotiation. Mean-

while, the platform will provide its server certificate for the client to verify. This mechanism allows the mutual authentication that is going to be carried out in the negotiation phase.

Once the negotiation phase has been finished successfully, the server presents customized contents to the user. As the user presents a valid certificate, the server identifies the user roles and according to this, prepares the layout. Not registered users are considered new clients and are registered as such during the first negotiation phase.

Once this phase is concluded, the user may request a digital copy of an offered product. At this stage the *protection* phase takes place. During this phase, the system must provide a protected copy of the requested product to the client. So, the WS receives the request to download a copy of a product identified by a hash. The platform makes use of hash identifiers in order to avoid inference of other product identifiers. At this time, the WS checks internally if the user already purchased a copy of the requested film, if so, the WS contacts the FS to get the already marked copy of the product. This means that a user that owns a unique certificate, can only have one marked copy of each original.

If the WS does not find a copy already purchased by the requester, it will contact the TS to request a new copy. The TS generates then a new copy by first, generating a new mark and secondly by asking the MS to embed the given watermark in the original product, generating a unique copy that the TS will then assign to the requester. At this time the WS finds as a result a marked copy assigned to the user.

The *delivery* phase starts when the TS notifies the WS that a new copy has been assigned to the requester, passing the copy hash to the WS as well as the location of the FS where the copy is located. The WS contacts the FS to download the copy and forwards it back to the user.

Note that, any interaction between services is carried out using the service certificates hosted on each service instance. Hence, the WS contacts the TS, and both perform a mutual authentication before the transaction is carried out. Every time a connection is established, both ends verify each other's certificate against the CA OSCP service [8].

4.1 Platform Functionalities

What has been presented above consists on the basic flow of a product request. Moreover, the platform provides a set of functionalities for the different actors that may interact with the system in order to be able to perform the flow presented in the previous section. Indeed, the platform proposes four different roles:

- **Administrator**: manages and configures the platform through the administration functionalities.
- **Client**: requests new digital contents and navigates through already downloaded resources.
- **Guest**: navigates through published contents but cannot download them.
- **Content manager**: manages the digital contents and stocks.

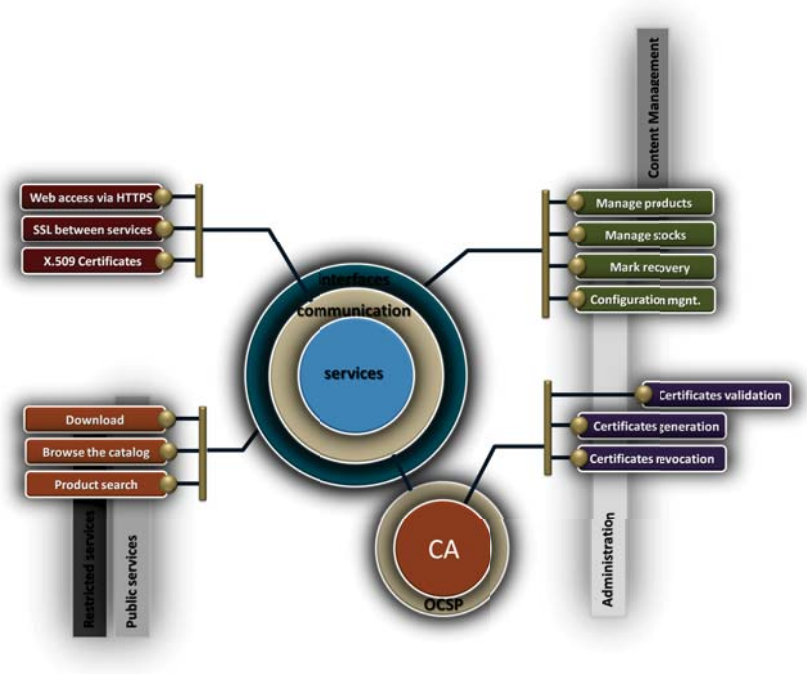


Figure 2: Graphical representation of platform main functionalities.

Figure 2 presents the functionalities available for each of the proposed roles. Hence, users with administrator's role have full access to the administration functionalities;

- **Mark recovery**: Allows the extraction of the mark embedded in a digital content. Via a web form, the user submits the digital copy with an embedded mark and specifies, from a list of products, the name of the original content that the copy was made from. As a result, the user obtains a link to the copy managed internally, together with information about the registered user to whom the copy was assigned.
- **Configuration management**: Allows the user to adequate the platform to the deployment context. There exist static and dynamic parameters that can be modified before and after the platform is started-up, therefore affecting the platform behavior.
- **Certificate generation**: As mentioned previously, any actor of the system must own a valid digital certificate generated by the certification authority (CA). By means of this functionality, the user can generate new certificates for new clients or new server instances. This functionality is offered directly by the CA through the Web interface using the proper client certificate.

- **Certificate revocation**: In some circumstances, under a security compromise, it may be recommended to revoke certain certificates. The CA offers a Web interface for that purpose. Considering that all platform services verify client certificates on each request, a certificate revocation has an immediate effect, offering fast reactivity to potential security vulnerabilities.

On the other hand, users with content management's privileges have access to the following functionalities;

- **Manage products**: The platform offers a complete Web interface to manage multimedia contents. A user with proper privileges is able to navigate through all the products in the catalog browser, search for specific products via the search form, modify certain products' attributes and upload new contents.

- **Manage stocks**: Although stock management will be introduced later, a user with content manager privileges has access to the Web interface to check the products stock level. By knowing the actual stock level for each product, and analyzing the latest client interests, the content manager can generate new copies of certain products by modifying its minimum stock level.

The last set of functionalities are available to those clients with (Restricted services) or without (Public services) a valid user certificate.

- **Browse catalog**: Any user consulting the web site, having or not a user certificate, can visualize the catalog browser that presents available categories. Categories are defined and maintained by the content manager and serve to group the movies according to its typology. For instance, the platform defines twelve categories. Among others we have: horror, adventure, animation, drama, comedy, etc. The catalog, located on the right column, shows all categories with the number of products available. By following one of the categories link, all products belonging to it are presented.

- **Search products**: The platform provides all users, having or not having a user certificate, with a fast search form. Through this fast form, the user may enter some key text related to the product he/she is looking for. The products matching the search criteria are presented.

- **Download products**: The download functionality is only available to registered users, that is, those having a valid user certificate. So, after a user has found the desired product, either via the search form or the catalog browser, a request can be made via the download link. A copy of the original product, with an embedded mark will be returned. Successive downloads of the same product return the same copy, stored in the platform repository.

4.2 Stock Management

Considering that the main functionality of the copyright protection platform is to provide marked digital copies, the management of the product life cycle is a very important

aspect to consider. The content manager is responsible for the product life cycle management, providing the original copies of the video files as well as the support attributes that serve to define the product: name, description, video cover, categories and minimum stock level. As already mentioned earlier, when a user requests a new product from the catalog, a watermarked copy is provided. But the watermarking process is not immediate, therefore, if the system generates copies of the original product on-the-fly, the user would be waiting for too long. As an order of magnitude, watermarking a DVD quality sample with a bit rate of 7500 kb/s and a duration of 110 minutes (video stream size of 4,89GB) on an Intel(R) Core(TM) Xeon E5645 CPU @ 2.40GHz takes 17 minutes using a single thread and 10 minutes when using 4 threads, long enough to avoid having the user waiting for his/her brand new purchase.

To avoid the generation of copies based on user requests, the platform generates a given number of copies when the content manager uploads a new product. So, according to the "minimum stock level" attribute of the new product, the system generates this precise number of marks and embeds them into the same number of copies.

Later, when a user requests a copy of a given content, the platform identifies a non assigned copy. Such copy is then assigned to the requester and returned as an attachment. Afterwards, a stock level verification is performed on all existing products. The stock level of a particular product consists on the number of generated copies not yet assigned to any user. When the stock level of a product reaches the minimum defined as a parameter on a product upload, new copies are created as to reach such level. On the other hand, the content manager has the ability to control the stock level manually. Therefore, the minimum stock level of certain products can be modified to satisfy a specific user demand.

Although the system architecture will be presented in following section, the marking service consists on a number of modules that can provide copies of original products concurrently. More over, according to this design principle, if multiple users request the same product, and possibly deplete the stock, the system will generate new copies in order to regenerate the stock level by launching multiple requests to available watermarking services, and thus satisfying peak demand points.

4.3 System Architecture

The proposed platform has been designed and developed as a distributed JEE application. The platform is basically made up of two software modules; web modules and service modules. Figure 3 shows the basic components that conform each module.

Both modules are build up around the Spring 3 JEE framework [3] which powers up the development cycle of Web applications. Moreover, the database repository is accessed via Hibernate [4] components which are published as a JAR library. The Object/Relational (O/R) mapping components consist on the basic Plain Old Java Objects (POJO) that map the relational database tables to Java objects using Hibernate Java Persistence Annotations (JPA). The upper layer offers a set of Java objects to facilitate the POJO management, the so called Data Access Objects (DAO).

[3] http://www.springsource.org/
[4] http://www.hibernate.org/

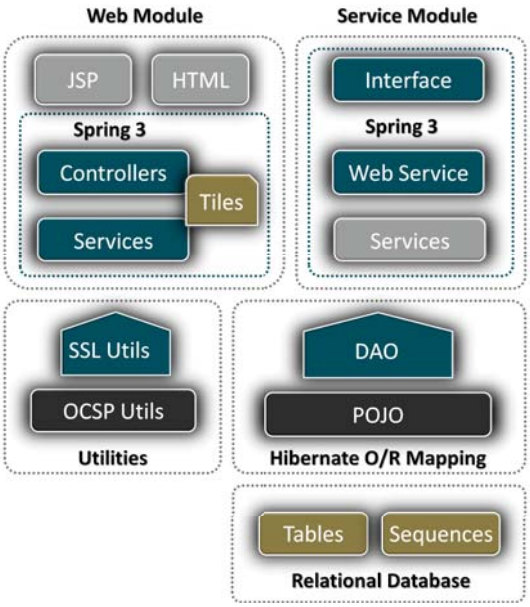


Figure 3: Diagram of the generic platform modules

The Web module consist on a set of controller components in charge of managing the user requests, each controller maps a set of URLs the user may follow when accessing the Web portal. The controller dispatches the business logic execution to the proper services, when concluded, the services provide information in the form of POJO that are rendered dynamically by Java Server Pages (JSP). The whole view layer is build up using Spring 3 tiles, which facilitates the graphical design and maintenance of the Web portal.

Although the service module architecture is very similar to the web module's, the interface changes from HTTP/HTML to HTTP/SOAP [9]. This way, the platform services publish their functions externally as standard Web Services. All platform components are packaged as Web archives (WAR) and can be deployed on any servlet container.

The available type of services were slightly introduced on the preliminary sections, indeed, the platform offers, besides the TS, a File Server (FS) and a Marker Server (MS). Although there is a single instance of the TS, there may be multiple instances of FS and MS services depending on the system needs. The platform proposes a service registration process in order for a process to be active and public.

So, when a service starts-up, the first thing it does is to contact the TS to register on duty. When the connection is established between the new candidate and the TS, and by means of mutual authentication, the TS validates the candidate certificate against the CA, as explained in the previous section. If the validation is successful, the TS registers the service in a UDDI [10] (Universal Description Discovery and Integration)

repository. This will allow the TS to query for active services later. When a service closes in a controlled manner, the de-registration process takes over, eliminating the service from the UDDI repository.

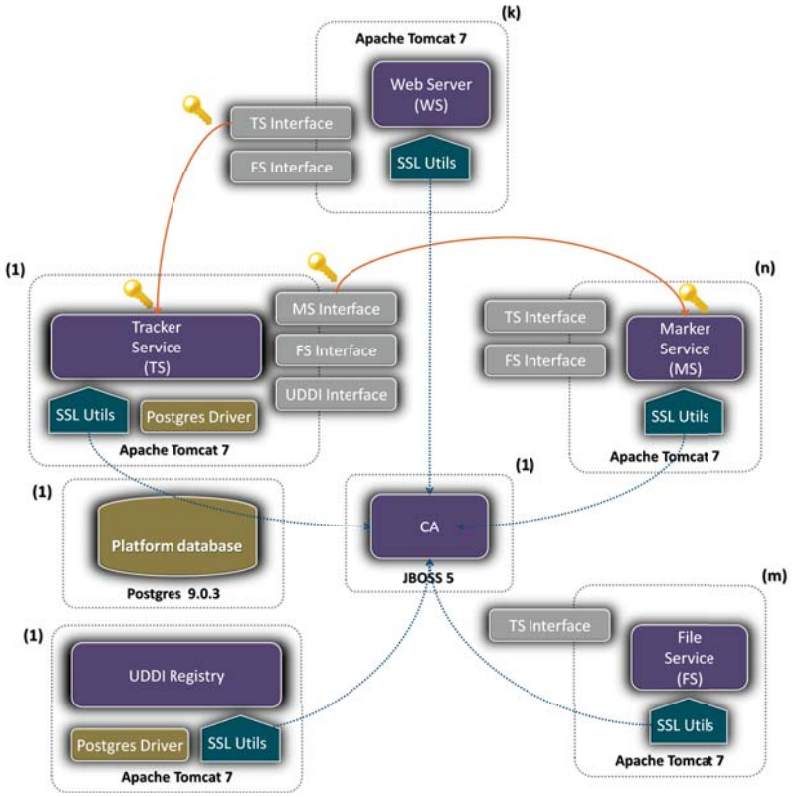


Figure 4: Deployment diagram of the services proposed in the platform

Figure 4 shows the deployment diagram of all components considered in the platform. Each dashed box represents a deployment container, indeed there are two different containers; Apache Tomcat 7 [5] and JBOSS 5 [6]. Tomcat holds all platform services while JBOSS holds the Certification Authority (CA) which is implemented using an Open Source PKI Certificate Authority called EJBCA [7].

The figure shows as well the communication protocols used between the components. Basically, the figure shows only two communication protocols; OCSP and SOAP. OCSP, which stands for Online Certificate Status Protocol, and it is represented as a blue dashed line, is carried out between all platform services and the CA to check the validity of a given certificate.

[5] http://tomcat.apache.org
[6] http://www.jboss.org/jbossas
[7] http://www.ejbca.org

The rest of interactions between services are carried out using the SOAP protocol. For simplicity, the figure shows only two orange lines representing the SOAP communication between services. Moreover, the representation of the public interfaces in the Web Service clients helps infer the rest of missing lines. As an example, on certain user requests, the WS contacts the TS via the TS Interface, this interface consists on the Java classes generated at client side given the server's WSDL [11] (Web Service Definition Language). Therefore, any module holding a TS Interface requires an orange line between the interface and the TS. The TS delegates the task of marking to the MS, which is done via the MS Interface.

Note as well that all communication carried out under SOAP protocol requires both client and server valid certificates. This allows for mutual authentication and data encryption in both directions.

The figure represents as well the number of permitted instances for each service, represented as a number inside brackets. As mentioned before, the number of FS (m), WS (k) and MS (n) instances can be configured depending on the context scenario, and it has to be done at deployment time.

4.4 Platform User Interface

The figure 5 shows the user interface presented to an authenticated user when first login. The left figure 5(a) presents a set of the most popular product covers. The cover image link redirects to the product detailed information 5(b). Below the cover flow, all products are presented, by default, in name alphabetical order. The sort criteria can be modified through the combo options located on top of the products list.

(a) Most popular product covers (b) Product detailed information

Figure 5: System welcome page and product detailed information view

The product information page presents detailed information not available on the products list page (Figure 6). Depending on the user role, the page presents basic or full attributes. The basic attributes are available to clients and provide information about the product itself, such as the name of the film, a description, the categories it

belongs to, the author, the publication date, the purchase date in case already has been purchased and the product size.

The administrator role can visualize extra product attributes; minimum stock level, product sequence number (considered to identify the next fingerprint to embed on the next copy), the DCT matrix positions where the watermark is inserted, distance between marked frames, coding, mark depth, and specific coefficients for watermark generation. Moreover, if the user has the content manager role, the view provides a link to modify the product attributes.

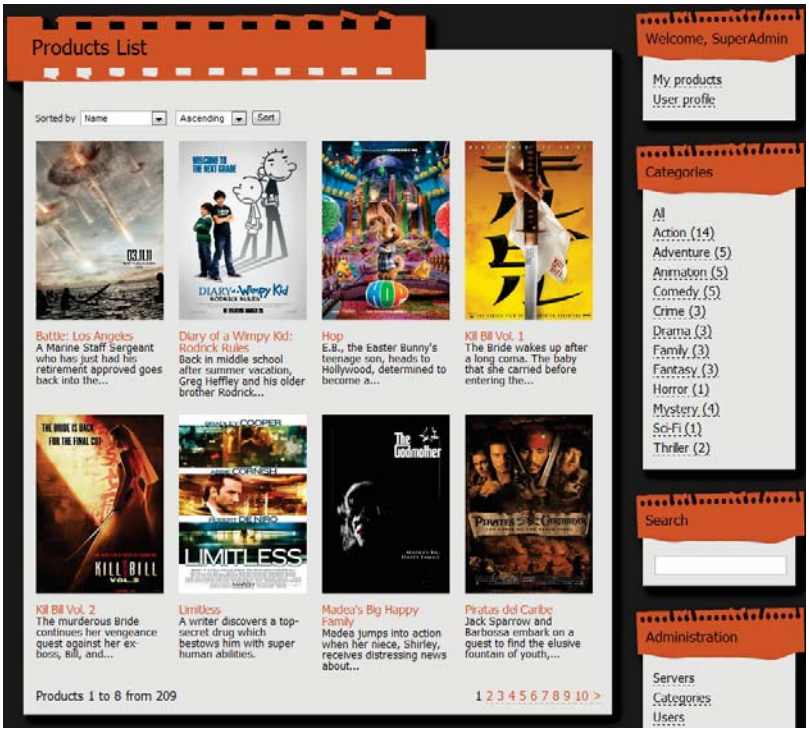


Figure 6: Products list and available menu options (right side)

The Web page displays the menu options grouped in four categories, as presented in Figure 6; user specific options, categories, search and administration. Obviously, depending on the user role, some of the options may be hidden, as are specific for administrators or content managers. For instance, a basic user or a guest has access to user specific options, categories and search options. On the other hand, administrator and content manager roles have access as well to the administration group.

The administration menu provides a set of options for the platform management. The following is only available for administrators and content managers:

- **Servers**: this option provides the administrator with a list of all available servers offering a platform service, that is, a tracker service, a marker service or a file

system service. The list of services is extracted from the UDDI registry and presented as a plain list with some extra attributes. Through this list, the administrator can check the number of existing services of a given type.

- **Categories**: the content manager can update the list of categories through this functionality. Categories can be extended or modified to satisfy the available offer. Categories are offered to the content manager to classify new products or to update existing ones.

- **Users**: the administrator can, through this view, take a look at the list of registered users. This list presents the user certificate name, the serial number and the state (whether the user is active or not). Through this list, the administrator may cancel a user account by de-activating it, this will trigger a certificate revocation as well as an account cancelation. The following user login attempts with a deactivated certificate will be denied.

- **Parameters**: Some of the services activity can be parameterized, therefore adapting the system to certain context conditions. These parameters can be managed through this view. There are parameters related to the watermarking process as well as the Web interface layout.

- **Stock**: as already explained in Section 4.2, the platform provides a functionality for the manual management of stocks. Through this link, the content manager visualizes the list of products with the number of available watermarked copies as well as the minimum configured stock level. At any time, the minimum stock level can be updated to force a later stock regulation.

- **Upload**: the content manager has the responsibility of providing new content by uploading, describing and categorizing multimedia contents. This link offers a Web form to define the characteristics of the new product that will be available to registered users for download.

- **Mark recovery**: in order to ensure copyright protection the administrator and/or content manager can use a tool to recover the watermark that has been embedded on a product copy. This link provides access to a Web form where the user introduces the recovered copy location and the name of the original product the copy was made from. The result of the analysis process consists on the list of users involved in the dishonest copy distribution. There may be more than one dishonest user if a confabulation process has taken place.

4.5 Integration with payment methods

Even though in the platform it hasn't been integrated with any payment method, it has been taken into account the use of 2 novel payment schemes. The main idea of the business model is to make the platform accessible by means of public-access video systems (for instance, in the waiting-rooms of the airports). In this scenario, a user wants to see a film (or a part of it) while she is waiting for boarding. Moreover, she wants to pay by means of her mobile device equipped with a Near Field Communication.

According that, the platform can be adapted to work together with the proposals introduced below.

4.5.1 Secure e-ticketing payment scheme for mobile devices with NFC that includes exculpability and reusability

An electronic ticket is a contract, in digital format, be- tween the user and the service provider, and reduces both economic costs and time in many services such as air travel industries or public transport. However, the security of the electronic ticket has to be strongly guaranteed, as well as the privacy of their users. The proposed electronic ticketing sys- tem considers these security requirements and includes the exculpabil- ity as a security requirement for these systems, i.e. users and the service provider can not falsely accuse each other of misbehavior. The system en- sures that either both parties receive their desired data from other or neither does (fair exchange). Another interesting property is reusability. Thanks to reusability the tickets can be used a predefined number of times with the same security as single tickets. Furthermore, this scheme takes special care of the computational requirements on the users side by using lightweight cryptography. We show that the scheme is usable in practice by means of its implementation using mobile phones with Near Field Communication (NFC) capabilities. More details about this proposal could be find in the appendix C.

4.5.2 A Secure Automatic Fare Collection System for Time-based Services with Revocable Anonymity for Users

Automatic Fare Collection (AFC) systems calculate the fare that the users must pay depending on the time of service (time-based) or the points of entrance and exit of the system (distance-based). The progressive introduction of Information and Communication Technologies (ICT) allows the use of electronic tickets, which help to reduce costs and improve the control of the infrastructures. Nevertheless, these systems must be secure against possible fraud and they must also preserve users? privacy. Therefore, we have studied the security requirements for the time- based and distance-based systems and, we have proposed a protocol for each of the AFC systems. The protocols offer strong privacy for honest users, i.e., the service provider is not able to disclose the identity of its users and, moreover, different payments of the same user are not linkable. However, anonymity for users could be revoked if they misbehave. The protocols have been implemented in the Android mobile platform and its performance has been evaluated in two Android smartphones. The results remark that protocols are suitable to use on AFC system with a medium class mobile device although they offer a better experience with a high-class smartphone. The appearance in the market of more powerful mobile devices suggests a better usability of our proposal in a near future. More details about this proposal could be find in the appendix D.

5 Conclusion

This article has presented the implementation of a platform for the delivery of copyright protected digital content. The copyright protection is based on a combination of watermarking and fingerprinting techniques that allow the generation of protected copies that can be distributed in a trusted environment. The dishonest redistribution of the purchased contents can be traced by means of watermarking extraction and fingerprinting analysis. Both techniques result in the identification of the malicious users involved in the unauthorized redistribution process. The solution is presented as a distributed JEE application based on standardized frameworks such as Spring and Hibernate. As a result the system offers loosely coupled Web services with a high cohesion. The independence and modularity of the services invite for good maintainability and high scalability. The platform presents a demonstrable evidence of a copyright protection system that proves the feasibility of the implementation of watermarking and fingerprinting algorithms on a real life scenario.

Moreover the platform has taken into account the use of 2 novel payment schemes. These payment methods permits the users to pay by means of mobile devices equipped with a Near Field Communications for the part of film that they have whatched.

A Watermarking Layer Technical Details

In this appendix we present the Secure Spread Spectrum techniques used to embed the marks into the multimedia content. Moreover, the modifications over the original algorithm is explained in section A.2.

A.1 Secure Spread Spectrum

Even though a lot of image watermarking systems exist, and some of them show better performance than secure spread spectrum, the algorithm presented in [2] by *Cox et al.* has been used in our implementation because it is easy to combine with a fingerprinting code and it offers a good protection against the typical distortions produced by video transcoding. Moreover, the computational cost of this algorithm, which is an important issue to take into account, is smaller than the computational cost of algorithms based on informed embedding or informed coding. In addition, as discussed in [2], Spread Spectrum offers enough robustness against scaling, transcoding and compression, which are the main processes performed during malicious video manipulation.

Spread Spectrum watermarking is an example of the spatial embedding of watermarks. The watermark is modulated by a pseudo-noise generator in order to produce a spread spectrum signal, which is then scaled according to the required power. The modulated signal is then added to the original image to produce the watermarked image.

To detect the watermark, either a high pass/edge detection or a Wiener filter is applied to the watermarked image to remove irrelevant information. The output of the filter is then correlated with the modulating pseudo-noise signal used at the transmitter side and compared to a predetermined threshold for the detection of the watermark. In

general, most watermarking techniques are considered a variation of the spread spectrum technique.

In [2], the mark $X = \{x_1, \cdots, x_n\}$ is embedded into $V = \{v_1, \cdots, v_n\}$ to obtain $V' = \{v'_1, \cdots, v'_n\}$. In that paper, V is a vector that contains all DCT matrix coefficients to be marked. The three different ways for computing V' presented in [2] are

$$v'_i = v_i + \alpha x_i \tag{2}$$

$$v'_i = v_i(1 + \alpha x_i) \tag{3}$$

$$v'_i = v_i(e^{\alpha x_i}) \tag{4}$$

Equation 2 is always invertible, and 3 and 4 are invertible if $v_i \neq 0$.

At the decoder, X^* is extracted from V^* and V. In the original proposal, the similarity of X and X^* is computed in order to measure how similar are the extracted and the original marks. A minimal value of similarity is defined as a threshold T to guarantee that no false positives are present. The similarity is defined as

$$sim(X, X^*) = \frac{X^* \cdot X}{\sqrt{X^* \cdot X^*}}, \tag{5}$$

and X and X^* will be considered the same mark if

$$sim(X, X^*) > T. \tag{6}$$

A.2 Modified-Cox Algorithm Implementation Details

The watermarking process performed by this sequence generator is slightly different from the one explained above and it is based on Equation 2. Essentially, the α value takes into account the quantization process which is applied to each coefficient. Formally, the performed modification is

$$v'_i = v_i + \alpha Q_i x_i \tag{7}$$

where Q_i is the quantizer matrix value, α is a strength factor, and x_i is the mark in polar format, that is +1 for a value of 1 and -1 for a value of 0.

Usually there are more positions suitable to be marked than bits to insert, the mark is repeated along all markable positions on the sequence. Because of this repetition the embedding system has a behavior similar to a repetition code.

Finally, the extraction is done by means of comparing the original video with the received one. If the difference is greater than a given threshold, we will consider that the originally embedded value was 1. If this difference is lower than another threshold, we will consider -1 as the original value. In any other case, 0 will be used as the original value. A value of 0 assumes that the value has been erased by the attacker.

B Fingerprinting Layer Justification

In this appendix we discuss the construction of codes presented in [5] and its decoding algorithm.

B.1 Background on Coding Theory

A subset C of a vector space $\mathbf{F}_q^n$ is called a *code*. The set of scalars $\mathbf{F}_q$ is called the *code alphabet*. A vector in $\mathbf{F}_q^n$ is called a *word* and the elements of C are called *codewords*, each codeword is of the form $\mathbf{x} = (x_1, \ldots, x_n)$ where $x_i \in \mathbf{F}_q$, $1 \leq i \leq n$.

The number of nonzero coordinates in $\mathbf{x}$ is called the *weight* of $\mathbf{x}$ and is commonly denoted by $w(\mathbf{x})$. The *Hamming distance* $\mathbf{d}(\mathbf{a}, \mathbf{b})$ between two words $\mathbf{a}, \mathbf{b} \in \mathbf{F}_q^n$ is the number of positions where $\mathbf{a}$ and $\mathbf{b}$ differ. The distance between a word $\mathbf{a}$ and a subset of words $U \subset \mathbf{F}_q^n$ is defined as $\mathbf{d}(\mathbf{a}, U) := \min_{\mathbf{u} \in U} \mathbf{d}(\mathbf{a}, \mathbf{u})$. The *minimum distance* of C, denoted by d, is defined as the smallest distance between two different codewords.

A code C is a *linear code* if it forms a subspace of $\mathbf{F}_q^n$. A code with length n, dimension k and minimum distance d is denoted as a $[n, k, d]$-code.

If we take a set of n distinct elements $P = \{\nu_1, \ldots, \nu_n\} \subseteq \mathbf{F}_q$, then a *Reed-Solomon code* of length n and dimension k, consists of all codewords of the form $(f(\nu_1), \ldots, f(\nu_n))$ where f takes the value of all polynomials of degree less than k in $\mathbf{F}_q[x]$:

$$\mathrm{RS}(P, k) = \{(f(\nu_1), \ldots, f(\nu_n)) \mid f \in \mathbf{F}_q[x] \wedge \deg(f) < k\}$$

A *simplex code* or *dual binary Hamming code* S_r, is a $[2^r - 1, r, 2^{r-1}]$ code, consisting of $\mathbf{0}$ and $2^r - 1$ codewords of weight 2^{r-1}, with equidistant pairs of codewords.

For any two words $\mathbf{a}$, $\mathbf{b}$ in $\mathbf{F}_q^n$ we define the *set* of *descendants* $D(\mathbf{a}, \mathbf{b})$ as

$$D(\mathbf{a}, \mathbf{b}) := \{\mathbf{x} \in \mathbf{F}_q^n : x_i \in \{a_i, b_i\}, 1 \leq i \leq n\}.$$

One can see that among the set of descendants of $\mathbf{a}$ and $\mathbf{b}$, there are $\mathbf{a}$ and $\mathbf{b}$ themselves.

For a code C, the *descendant code* C^* is defined as:

$$C^* := \bigcup_{\mathbf{a} \in C, \mathbf{b} \in C} D(\mathbf{a}, \mathbf{b}).$$

B.2 Construction of a Concatenated Fingerprinting Code

The idea of using code concatenation in fingerprinting schemes to construct shorter codes, was presented earlier in [1].

A concatenated code is the combination of an *inner* $[n_i, k_i, d_i]$ q_i-ary code ($q_i \geq 2$) with an *outter* $[n_o, k_o, d_o]$ code over $\mathbf{F}_{q_i^{k_i}}$. The combination consists in mapping the codewords of the inner code to the elements of $\mathbf{F}_{q_i^{k_i}}$, that results in a q_i-ary code of length $n_i n_o$ and dimension $k_i k_o$. Note that the size of the concatenated code is the same as the size of the outter code.

In [12], dual binary Hamming codes are proposed as fingerprinting codes. One of the major drawbacks of that scheme is that the number of codewords grows linearly with the length of the code. To overcome this situation in [5], code concatenation is used. The concatenation combines a dual binary Hamming code with an IPP Reed-Solomon code.

So, to construct a $[n(2^r - 1), r\lceil n/4 \rceil]$ binary fingerprinting code C, they propose:

- as inner code, a $[2^r - 1, r, 2^{r-1}]$ dual binary Hamming code S_r,

- as outter code, a $[n, \lceil n/4 \rceil, n - \lceil n/4 \rceil + 1]$ IPP Reed-Solomon code over $\mathbf{F}_{2^r}$,
- together with a mapping $\phi : \mathbf{F}_{2^r} \to S_r$.

The codewords of C are obtained as follows, take a codeword $\mathbf{x} = (x_1, \ldots, x_n)$ from the Reed-Solomon code and compute $\mathbf{y}_i = \phi(x_i),\ 1 \leq i \leq n$. The concatenation of the $\mathbf{y}_i$'s forms a codeword $\mathbf{y} \in C$, where,

$$\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n) \text{ such that } \mathbf{y}_i = \phi(x_i)$$

B.3 Overview of the Fingerprinting Concatenated Decoding Algorithm

The decoding algorithm proposed in [5] is done in two stages. First, we decode the inner code to obtain an n-tuple of sets of codewords. Then, with this n-tuple of sets we construct a reliability matrix that is used to decode the outer code.

Suppose we want to decode the following fingerprint:

$$\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$$

The inner decoding consists in the decoding of each subword $\mathbf{y}_i$ using a simplified Chase Algorithm. The output will be a single codeword $\{\mathbf{h}_1\}$, a pair of codewords $\{\mathbf{h}_1, \mathbf{h}_2\}$ or three codewords $\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3\}$.

Then, for $i = 1, \ldots, n$ we use the mapping $\phi(s_m) = \mathbf{h}_m$ to obtain the set $S_i^{(j)} = \{s_{i_1}, \ldots, s_{i_j}\}$, where the superscript $j \in \{1, 2, 3\}$ indicates the cardinality of the set. Note that the elements of the $S_i^{(j)}$'s are symbols from $\mathbf{F}_{2^r}$. We denote by $\mathcal{S}^{(1)}$ the set of the $S^{(1)}$'s, by $\mathcal{S}^{(2)}$ the set of the $S^{(2)}$'s and by $\mathcal{S}^{(3)}$ the set of the $S^{(3)}$'s.

We also define the n-tuple of sets $\mathcal{S} = (S_1^{(j)}, \ldots, S_n^{(j)})$, that is used to construct a reliability matrix. With this matrix we run a list decoding algorithm obtaining a list U of potential traitors. If d_o denotes the minimum distance of the outer code, then to extract the traitors out of the list U, we use the following statements:

- If $(|\mathcal{S}^{(1)}| + |\mathcal{S}^{(2)}|) > 4(n - d_o)$, then by Theorem 3 in [5], at least one of the traitors is identified with probability 1.
- If $|\mathcal{S}^{(2)}| > 2(n - d_o)$, then by Theorem 3 in [5], both traitors are identified with probability 1.
- If $(|\mathcal{S}^{(1)}| + |\mathcal{S}^{(2)}|) \leq 4(n - d_o)$, then define $U_3 = \{\mathbf{u} \in U : u_p \in S_p^{(3)},\ \forall\ S_p^{(3)} \in \mathcal{S}\}$. The only cases of positive identification are:
  - For any $S_p^{(2)} \in \mathcal{S}$, where $S_p^{(2)} = \{s_{p_1}, s_{p_2}\}$, if there are two and only two codewords $\{\mathbf{u}^1, \mathbf{u}^2\} \in U_3$, such that $u_p^1 = s_{p_1}$ and $u_p^2 = s_{p_2}$, then codewords $\mathbf{u}^1$ and $\mathbf{u}^2$ can be identified as traitors.
  - For any $S_p^{(1)} \in \mathcal{S}$, where $S_p^{(1)} = \{s_{p_1}\}$, if there is one and only one codeword $\mathbf{u} \in U_3$, such that $u_p = s_{p_1}$, then codeword $\mathbf{u}$ can be identified as a traitor.

C A secure e-ticketing scheme for mobile devices with Near Field Communication (NFC) that includes exculpability and reusability

Information technologies (IT) are becoming usual in our society as they progressively replace the use of paper in many of our common operations. An example of paper ticket could be the air flight boarding pass. Vodafone and Spanair [13] conducted an e-ticketing test in Spain (May 2007). The passengers received the electronic boarding pass in their mobile phone, and they were able to go directly to the security control area, and later board. The International Air Transport Association (IATA) started in 2004 a program to introduce the use of electronic tickets. IATA estimates that the no-use of paper tickets will reduce the costs by US$ 3000M [14] boosting disintermediation by using electronic tickets.

The electronic tickets have not been used only as a boarding pass. They can also be used in multiple transport services. The AMSBUS [15] booking system from the Czech Republic allows the purchase of SMS tickets. The passenger receives the ticket in her mobile phone; then, the user shows the message to the ticket inspector when needed. Leeds United [16] supporters can book one of their sport events and later receive an SMS with the booking confirmation together with some added information such as the assigned seat.

These examples show the progressive introduction of electronic tickets on different kinds of services and the increasing use of mobile phones in all of them as the most suitable e-ticket storage device. In addition to that, the real application of these electronic ticketing systems depends on their security, due to the ease of copy of electronic data. Electronic tickets have to keep the same security that is offered in paper tickets.

The main focus area of the present paper is the development of a secure e-ticketing scheme for mobile devices. Our protocol presents a fair-trading mechanism during the ticket verification in such a way the user pays in exchange of the right to use the agreed service. As in our previous work [17], the protocol is designed to meet the set of security requirements for such schemes described in Section C.1.1. We would like to highlight the exculpability property, which is a new property that we have first introduced in the e-ticketing schemes (i.e. the service provider can not falsely accuse the user of ticket overspending, and the user is able to demonstrate that she has already validated the ticket before using it). In addition to that, now the protocol has been enhanced with the property of reusability (i.e the tickets can be used a predefined number of times with the same security as single tickets). The final contribution of our paper is the implementation of the proposed e-ticketing protocol using light-weighted mechanisms by means of low computational complexity cryptography and low communicational overhead.

An analysis of the previous works related to our proposal is presented in Section C.1. Later, in Section C.2, our scheme is accurately defined. The security and privacy analysis of the scheme is detailed in Section C.3. Implementation details and performance results is given in Section C.4.

C.1 Previous work

First of all, in Section C.1.1, we start presenting the security requirements that have to be achieved in these systems, as well as we introduce a new security requirement that is not achieved in the previous works and which could be taken into account in the future: *exculpability* together with a property that we have recently achieved: *reusability*. Section 2.2 lists other relevant properties of e-ticketing systems. Once the requirements are described, the e-ticketing proposals have been classified in Section C.1.3.

C.1.1 Security requirements

E-ticketing systems have to consider and guarantee the following security requirements:

- Authenticity : A user has to be able to verify if an e-ticket has been issued by an authorized issuer.
- Integrity: A user has to be able to verify if his e-ticket has been altered as regards the one issued by the correspondent authorized issuer.
- Non-repudiation: Once a valid e-ticket has been issued, the issuer has not to be able to deny that he has issued that ticket with its contents. Observe that, in fact, this property comprehends the two previous properties: if the issuer can not deny having issued an e-ticket it means that integrity and authenticity properties have been achieved.
- Unforgeability: Only authorized issuers can issue valid e-tickets.
- Non-Overspending: E-tickets can only be used as agreed between the issuer and the user. Non-reusable e-tickets can not be reused after they have been spent (by the same or other users). Reusable e-tickets can be used exactly the number of times agreed in the moment of issue. Finally, some e-tickets can not be used after their valid period of time. Mechanisms to control overspending can affect the following property: anonymity. Overspending can be prevented or detected. If overspending is detected in the verification phase overspending will not be allowed. If it is detected afterwards, some way to identify the overspender or possible overspenders will be necessary. Some authors [18] call duplication to this property.
- Anonymity: Not all the paper-tickets present the same requirements related to anonymity, so we have to distinguish some possible scenarios for e-tickets. There are two anonymity degrees: non-anonymous (the service requires user identification and authentication) and revocable anonymous (the service is anonymous, but it can be revoked if the user misbehaves).
  - Non-anonymous e-tickets: Some e-tickets will have to be non-anonymous; it means that user identity has to be embedded in the e-ticket in some way, in order the service provider could verify that the user is authorized to spend the e-ticket. This is the case of plane e-tickets. In the boarding phase the

auxiliary staff of the Air Company have to be able to verify that the flyer identity is the same that the identity contained in the e-ticket.

  - Anonymous e-ticket: Some paper tickets allow users to remain anonymous in front of the issuer and/or the verifier. Therefore e-tickets will have to maintain the property. Perhaps in the issue phase the user is identified (it depends on the kind of payment used), but that payment has not to be linked to the issued e-ticket. In any case, it has to be able to spend the e-ticket without any kind of identification. Even colluded issuers and service providers should not be able to break anonymity of honest consumers.
  - Revocable anonymity: If overspending is detected after the verification process, it means that the same e-ticket could be used more times than desired (by issuer and/or service provider). For non-anonymous e-ticket this is not a problem: we know overspender user and so actions can be undertaken. For anonymous e-tickets, anonymity has to be revocable in order to identify overspenders. Obviously, fair users would have to remain anonymous or, at least, they would have to be able to prove they are fair users. Some e-ticketing schemes allow also the revocation of the anonymity of the user if he misbehaves using the service.

- Expiry date: A ticket could be only valid during a time interval.

- Online/Offline: Ticket verification can require a persistent connection with a trusted centralized system (online); otherwise, that connection is never necessary (offline).

- Fairness: During the execution of an e-ticketing protocol the parties execute several exchanges of elements. One of these exchanges is produced during the issue phase. Many times an e-ticket is exchanged for a payment or some other element. We can think of some exceptions: donations (between users), free e-ticket (for some events), etc. But in the general case user will have to pay for an e-ticket. During the validation phase another exchange is produced between the user and the provider: the ticket and the service (for that reason they can exchange exculpability proofs). Therefore a protocol for that exchange will have to be designed, and some properties achieved. We are in front of a kind of fair exchange of values (an e-ticket for a payment, a serveice for an e-ticket), and so, some of the following properties will be necessary: fairness, abuse-freeness, timeliness, verifiability of the TTP, etc. ItÕs out of the scope of this paper to explain these properties that can be found, for instance, in citefer09.

- Transferability: Some paper tickets can be transferred to other people (spectacle tickets, bus tickets, etc.). Obviously it is not the case of identified e-tickets (plane e-tickets, etc.). People receiving an e-ticket in a transfer (not directly from an authorized issuer) has to be able to verify that this e-ticket is valid (it will be easy if non-repudiation, integrity and authenticity are met) and not spent by the transferring entity. When we are in front of gifts or donations between confident people (a friend, familiar, etc.) no special measures have to be taken, it's a personal matter if afterwards an overspending occurs. But perhaps e-tickets can be

resold, or e-tickets (spectacle entrances) can be a ÒpresentÓ from a third company (in exchange of buying some product from this company). The receiving entity has to be sure that the e-ticket is valid and not spent. But itÕs possible that the user will try to overspend the e-ticket, and the transferring entity has to be able to prove he has not re- used the e-ticket. This problem should be specially handled when anonymity is revocable. Transferability will make necessary the fairness property.

Exculpability None of the analyzed proposals deals with exculpability; that is, the service provider can not falsely accuse the user of ticket overspending, and the user is able to demonstrate that she has already validated the ticket before using it. The exculpability is an interesting issue in our case, because the e-ticketing scheme should ensure that either both parties (users and provider) receive their desired data (e-ticket and the validated e-ticket) from other or neither does (fair exchange). The parties agree to reveal its data only if the other part also agrees. If any party deviates from the scheme then it can be identified as the culprit by the Trusted Third Party (TTP). Our scheme defined in Section C.2 takes exculpability as a security requirement for an e-ticketing system, as a first step to include this security requirement in future works.

Reusability A ticket could be used once (non-reusable) or many times (reusable). In both cases, ticket overspending has to be prevented. Tickets can be used more than once as is the case of some urban transport, where a transport pass can be used for several travels (and a counter is decreased in every travel) or it can be used over a period of time. Even, sometimes, the same ticket can be used in different places (for instance, bus and underground in the same city). E-tickets have to incorporate security measures that allow using the ticket in the valid period of time or for the number of uses agreed (or a combination of both, time and uses). Some authors name divisibility to this property (probably influenced by the similarities between e-ticket and e-money).

C.1.2 Other requirements

There are some other requirements that they are not so directly related to security, but they can be so important than those explained previously.

- Portability: E-tickets, as paper tickets, have to be portable by users. So, it has not to be necessary a laptop or a personal computer to handle e-tickets. Mobile phones, smart cards, etc. will have to be able to store and process e-tickets.
- Reduced size: Typically, e-tickets will be stored in mobile devices (a mobile terminal as a mobile phone, a smart card, etc.), and sometimes these devices will have a limited memory. Therefore, e-tickets have to be reduced in size as possible.
- Flexibility: We can think of a lot of different tickets (plane tickets, bus tickets, concert tickets, museum tickets, etc.). We can design a specific e-ticket for each application, or we can adapt a general e-ticket for each application. Obviously

the later solution is preferred in order to economize the solution, and it will allow a better security analysis.

- Ease of use: We are thinking e-ticketing as a solution for general public (using paper tickets nowadays, and not necessary especially confident in electronic means). E-tickets have to be as easy to use as paper tickets, and without new problems for users.

- Efficiency: We can think efficiency from two points of view. First, mobile terminals can be limited in terms of computational power, and so protocol operations and especially cryptographic operations have to be reduced only to necessary ones. Second, communication capacity can also be limited, and so the protocol has to be designed with this constrain in mind. Whatever, the delay due to verification of validity of e-ticket has to be reasonable to be a valid solution for ticketing by electronic means.

- Payment system: The design of an e-ticketing system has to bear in mind that sometimes it will be necessary a payment system to obtain the e-ticket. So, e-ticketing system has to allow different payment systems to be used in order to pay for the e-ticket (if necessary).

- Globally-spendable: Costumer should be able to spend their e-tickets at any appropriate service provider.

- Offline verification: In some scenarios it will not be possible to contact with external databases or Trusted Third Parties, to verify if e-ticket is valid or not. Perhaps it will not be the general case, but a solution for this case has to be thought. This property is much related to the security mechanisms adopted. If an online solution is preferred, efficiency has to be especially remembered.

- Availability: This property can be seen as a security property, but it is quite difficult to address this problem only as a security one. We are thinking in denial of service attacks (difficult to handle), disaster events (more difficult to handle) or temporal malfunction of infrastructure (for instance, a power failure). This can mean that e-tickets can not be verified, and sometimes the event can not be delayed (a concert, a plane, etc.). A procedure to handle these situations has to be designed.

C.1.3 Classification of proposals

In this section, the proposals have been differentiated by the devices used in the systems: the *smart-card-based*, and the *non-smart-card-based* ones, with a deeper analysis performed in the non-smart-card-based proposals, taking into account their anonymity compliance.

Smart-card-based Smart-card-based proposals [19, 20, 21, 22, 23, 24, 18] establish a communication channel with the verification system. Thus, the most sensitive operations are sent to the smart-card through this channel. The smart-card verifies each

operation, so that users can not perform any non-allowed action. Security of smart-card-based systems rely on the smart-card security. If the smart-card security is compromised, the security of the entire system is also compromised. One recent example is the case of the Mifare cards, where in [25] the authors succeeded to compromise them. As a result, all the entire public transport system that used exclusively Mifare cards was compromised.

Non-smart-card-based Non-smart-card-based systems [26, 27, 28, 29, 30, 31, 32] allow to perform applications with higher computation requirements while taking advantage of their high storage capacity and also their wireless short-range communication resources; this is the case of the mobile phones, smart phones or PDA's. However, as these devices are not considered tamper-proof devices, e-ticketing systems require then high-level cryptographic protection in order to assure that users follow the e-ticketing protocol correctly. We classify the non-smart-card-based systems depending on *non-anonymous*, *anonymous* and *revocable anonymous* compliance.

Non-Anonymous proposals Some proposals are oriented to services where anonymity can not be provided to the user, or simply, these systems are not conceived to achieve anonymity at all. Between the proposals that do not consider anonymity, digital signatures are commonly used [27, 28].

In [28], either the user or the e-ticket should be identified in order to prevent problems such as malicious attacks. There is a real relationship between anonymity and transferability for user and e-ticket identification and reusability is also considered for other ticket information, such as its destination. Online mode is used in this scheme for security reasons, as they say offline systems show weaknesses to malicious attacks.

Anonymous proposals Haneberg et al. [29] present an electronic onboard ticketing scheme, by using a PDA connected to the system through Bluetooth and using Java for all applications. PDAs are chosen for their short-range wireless communications and the display. Anonymity is achieved in this proposal as no personal data is needed, and anonymity then only depends on the payment method used.

In [30], Quercia and Hailes' e-ticketing system proposal is based on Chaum's e-cash blind signatures, providing anonymity to the user, but the communication cost could be high, and possibly slow down the system. Apart from anonymity, non-repudiation, offline verification as well as portability are achieved in this proposed system.

The great majority of the described proposals that comply with anonymity are based on Chaum's blind signatures [33].

Revocable-Anonymous proposals In the proposal of Patel and Crowcroft [26] the security requirements are defined, where revocable anonymity is achieved, as well as offline mode, although central authority intervention is needed in order to prevent overspending.

Depending on the services, anonymity, transferability or reusability would be required in the Fujimura et al. proposal [34]. Pseudonyms are proposed if anonymity is required, and overspending is controlled by a central database (online mode).

In [31], Heydt-Benjamin et al. made a proposal using latest advances in e-cash to improve privacy in electronic ticketing systems for public transit. It uses pseudonyms in order to achieve anonymity.
Chen et al. [32] propose the use of mobile devices (mobile phones, smart phones or PDAs) in e-ticketing systems, by taking advantage of their wireless communications. They focus on the compliance of several security requirements, as (revocable) anonymity, non-repudiation, as well as efficient verification. The ticket process is defined in 3 phases in the paper: request, issue and verification. Anonymity is achieved by the use of pseudonyms.

The majority of the studied proposals use pseudonyms in order to achieve revocable anonymity. If pseudonyms are used, real identity information is not put into the ticket, only its pseudonym. But if the issuer could link every pseudonym to its real identity, then anonymity could be compromised. For that reason, only revocable anonymity for the user could be achieved. In this scenario, user traceability could be easily performed if user does not change its pseudonym regularly because the same pseudonym would be used for different tickets. Certain volume of data could allow all the involved participants to make user profiles if there were no pseudonym controls.

According to reusability there is a diversity of considerations; Patel and Crowcroft [26] believe that tickets can be reusable; Haneberg et al. [29], Heydt-Benjamin et al. [31] and Chen et al. [32] do not consider ticket reusability; finally, Quercia and Hailes [30] consider that reusability mainly depends on the service.

There is a remarkable equality between the proposals that use online verification [26, 31] and the ones which use offline verification [29, 30, 32]. Otherwise, ticket transferability is unanimously not considered in the proposals [26, 29, 30, 31, 32].

C.2 E-ticketing scheme

The scheme has the following actors: the user $\mathcal{U}$; the ticket issuer $\mathcal{I}$, who sends a valid ticket to $\mathcal{U}$; the service provider $\mathcal{P}$, who verifies the ticket and gives the service; and finally the TTP $\mathcal{T}$, who preserves $\mathcal{U}$'s anonymity, and gives a valid non-identity pseudonym to $\mathcal{U}$. The e-ticketing scheme has been designed for mobile devices, reducing the computation requirements in the user side, and providing the basic security requirements (authenticity, non-repudiation and integrity) together with expiry date, revocable anonymity, exculpability and reusability. The ticket verification is performed offline when there is only one provider for each service, although the scheme could be extended with multiple providers that offer the same service. In that extended case, the scheme would prevent overspending through online verification, requiring then the interconnection of the service providers which offer the same service. In any case, the connection with the issuer is never necessary. In Table 2, we define the details of our proposal, as well as some notation that is used in the scheme.

C.2.1 Phases

The phases of our system are: *Pseudonym Renewal*, where the user obtains a new temporal pseudonym to be used in the system without linkage to user's identity (if user behaves correctly); *Ticket Purchase*, that consists on the payment of the service and reception of the ticket; and *Ticket Verification*, where the user shows the ticket to the service provider in order to be checked and validated. Other phases considered in the system are claims. These claims should only be executed in case of controversial situations during the *Ticket Verification* phase: *Claim* m_2 *Not Received* (when $\mathcal{U}$ sends the first step of the verification m_1 but does not receive m_2 by $\mathcal{P}$, or the information is not correct);*Claim* m_3 *Not Received* (when $\mathcal{P}$ sends the second step of the verification m_2 but does not receive m_3 by $\mathcal{U}$, or the information is not correct); and *Claim* m_4 *Not Received* (when $\mathcal{U}$ sends the third step of the verification m_3 but does not receive m_4 by $\mathcal{P}$, or the information is not correct). Users have a digital credential ($\mathsf{Cert}_{\mathcal{U}}$) for authentication to the TTP only, as the system is anonymous, and all further movements in the system are tracked only with the assigned temporal pseudonym ($\mathsf{Pseu}_{\mathcal{U}}$).

Pseudonym Renewal The user $\mathcal{U}$ contacts the pseudonym manager $\mathcal{T}$ in order to renew the assigned pseudonym. The certificate $\mathsf{Cert}_{\mathcal{U}}$ identifies $\mathcal{U}$ through a secure connection established between the two parties. The system has the public parameters (α, p, q), where α is a generator of the group G with order p, being p and q large primes achieving $p = 2q + 1$. $\mathcal{U}$ generates a random value $x_{\mathcal{U}} \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes $y_{\mathcal{U}} = \alpha^{x_{\mathcal{U}}} \pmod{p}$ in order to receive a valid signed pseudonym $\mathsf{Pseu}_{\mathcal{U}}$ from $\mathcal{T}$. $\mathcal{U}$ and $\mathcal{T}$ have their own pair of keys used for signature and encryption of the transmitted data between them.

authenticateUser User $\mathcal{U}$ follows the next steps:

1. generates $x_{\mathcal{U}} \overset{R}{\leftarrow} \mathbb{Z}_q$, and computes $y_{\mathcal{U}} = \alpha^{x_{\mathcal{U}}} \pmod{p}$;
2. computes the signature $\mathsf{Sign}_{\mathcal{U}}(y_{\mathcal{U}}) = \mathsf{sk}_{\mathcal{U}}(\mathsf{h}_{y_{\mathcal{U}}})$ [8] where $\mathsf{h}_{y_{\mathcal{U}}} = hash(y_{\mathcal{U}})$ [9];
3. encrypts the information to be sent $(y_{\mathcal{U}}, \mathsf{Sign}_{\mathcal{U}}(y_{\mathcal{U}}), \mathsf{Cert}_{\mathcal{U}})$ with the $\mathcal{T}$'s public key as a digital envelope: $\mathsf{pk}_{\mathcal{T}}(y_{\mathcal{U}}, \mathsf{Sign}_{\mathcal{U}}(y_{\mathcal{U}}), \mathsf{Cert}_{\mathcal{U}})$ [10];
4. sends $\mathsf{pk}_{\mathcal{T}}(y_{\mathcal{U}}, \mathsf{Sign}_{\mathcal{U}}(y_{\mathcal{U}}), \mathsf{Cert}_{\mathcal{U}})$ to $\mathcal{T}$;

generatePseudonym Pseudonym Manager $\mathcal{T}$ executes:

1. decrypts $\mathsf{sk}_{\mathcal{T}}(\mathsf{pk}_{\mathcal{T}}(y_{\mathcal{U}}, \mathsf{Sign}_{\mathcal{U}}(y_{\mathcal{U}}), \mathsf{Cert}_{\mathcal{U}}))$ $\rightarrow (y_{\mathcal{U}}, \mathsf{Sign}_{\mathcal{U}}(y_{\mathcal{U}}), \mathsf{Cert}_{\mathcal{U}})$;
2. verifies $y_{\mathcal{U}}$: $\mathsf{pk}_{\mathcal{U}}(\mathsf{sk}_{\mathcal{U}}(\mathsf{h}_{y_{\mathcal{U}}})) \rightarrow (\mathsf{h}_{y_{\mathcal{U}}}) \overset{?}{=} hash(y_{\mathcal{U}})$;
3. if correct, then computes the signature of $\mathsf{Sign}_{\mathcal{T}}(y_{\mathcal{U}}) = \mathsf{sk}_{\mathcal{T}}(\mathsf{h}_{y_{\mathcal{U}}})$;
4. encrypts the signature with the $\mathcal{U}$'s public key: $\mathsf{pk}_{\mathcal{U}}(\mathsf{Sign}_{\mathcal{T}}(y_{\mathcal{U}}))$; and
5. sends $\mathsf{pk}_{\mathcal{U}}(\mathsf{Sign}_{\mathcal{T}}(y_{\mathcal{U}}))$ to $\mathcal{U}$.

[8] Note that $\mathsf{sk}_{\mathcal{E}}(content)$ means the decryption of *content* or the generation of a signature with its content *content* by using the private key of the entity $\mathcal{E}$

[9] Note that *hash*() is a public cryptographic one-way summarizing function that achieves collision-resistance. The notation $hash(item)^n$ is used to describe that the hash function is applied n times over the *item* (i.e. $hash(item)^n = hash(\overset{n}{\ldots\ldots}(hash(hash(item)))$

[10] Note that $\mathsf{pk}_{\mathcal{E}}(content)$ means the encryption of *content* or the verification of a signature *content* by using the public key of the entity $\mathcal{E}$

verifyPseudonym $\mathcal{U}$ computes:

1. decrypts $\mathsf{sk}_{\mathcal{U}}(\mathsf{pk}_{\mathcal{U}}(\mathsf{Sign}_{\mathcal{T}}(y_{\mathcal{U}}))) \rightarrow (\mathsf{Sign}_{\mathcal{T}}(y_{\mathcal{U}}))$;
2. verifies the TTP signature of $y_{\mathcal{U}}$: $\mathsf{pk}_{\mathcal{T}}(\mathsf{sk}_{\mathcal{T}}(\mathsf{h}_{y_{\mathcal{U}}})) \rightarrow (\mathsf{h}_{y_{\mathcal{U}}}) \stackrel{?}{=} hash(y_{\mathcal{U}})$;

Ticket Purchase The user establishes a connection with the ticket issuer $\mathcal{I}$ in order to receive the ticket. This connection could be established through an anonymous channel like TOR [35], guaranteeing then user's privacy. There are current contributions [11] that have implemented TOR for mobile devices with Android. $\mathcal{I}$ has a key pair and its public key certificate ($\mathsf{Cert}_{\mathcal{I}}$). Users do not use their personal keys (it would cause loss of anonymity); they use the temporal pseudonyms and authenticate through the Schnorr's Zero-Knowledge Proof (ZKP) [36]. The payment method is considered as out of scope in this proposal as we focus on the privacy given to user when joining/exiting the system, and using the service.
$\mathcal{I}$ generates the ticket with the information and its digital signature, together with the secret value $\mathsf{r}_{\mathcal{I}}$ and the secret shared key (they are decryptable only by $\mathcal{P}$ and $\mathcal{T}$) in order to let the provider show the secret value $\mathsf{r}_{\mathcal{I}}$ later, in the verification phase. The ticket issuer $\mathcal{I}$ and the user $\mathcal{U}$ follow this protocol:

getService $\mathcal{U}$ executes:

1. selects and pays for the desired service Sv;
2. generates a random value $\mathsf{r}_{\mathcal{U}} \stackrel{R}{\leftarrow} \mathbb{Z}_q$, and computes $\mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n)} = hash^n(\mathsf{r}_{\mathcal{U}})$, where n is the predefined maximum number of times that the e-ticket can be spent;
3. computes $\mathsf{H}_{\mathcal{U}} = \alpha^{r_{\mathcal{U}}} \pmod{p}$;
4. generates two more random values $a_1, a_2 \stackrel{R}{\leftarrow} \mathbb{Z}_q$ to be used in the Schnorr proof;
5. computes $A_1 = \alpha^{a_1} \pmod{p}$;
6. computes $A_2 = \alpha^{a_2} \pmod{p}$;
7. sends $(\mathsf{Pseu}_{\mathcal{U}}, \mathsf{H}_{\mathcal{U}}, A_1, A_2, \mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n)}, \mathsf{Sv})$ to the ticket issuer $\mathcal{I}$.

getChallenge $\mathcal{I}$ follows the next steps:

1. generates and sends a challenge $c \stackrel{R}{\leftarrow} \mathbb{Z}_q$ for $\mathcal{U}$;
2. asynchronously, for optimization, pre-computes ${y_{\mathcal{U}}}^c \pmod{p}$;
3. asynchronously, for optimization, pre-computes ${\mathsf{H}_{\mathcal{U}}}^c \pmod{p}$;

solveChallenge $\mathcal{U}$ computes:

1. computes $\mathsf{w}_1 = a_1 + c \cdot x_{\mathcal{U}} \pmod{q}$;
2. computes $\mathsf{w}_2 = a_2 + c \cdot \mathsf{r}_{\mathcal{U}} \pmod{q}$;
3. encrypts $(\mathsf{w}_1, \mathsf{w}_2)$ and sends it to $\mathcal{I}$: $\mathsf{pk}_{\mathcal{I}}((\mathsf{w}_1, \mathsf{w}_2))$;
4. pre-computes the shared session key used in the ticket verification: $\mathsf{K} = hash(\mathsf{w}_2)$;

getTicket $\mathcal{I}$ follows the next steps:

1. decrypts $\mathsf{sk}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}(\mathsf{w}_1, \mathsf{w}_2)) \rightarrow (\mathsf{w}_1, \mathsf{w}_2)$;
2. computes $\alpha^{\mathsf{w}_1} \pmod{p}$;
3. computes $\alpha^{\mathsf{w}_2} \pmod{p}$;
4. verifies $\alpha^{\mathsf{w}_1} \stackrel{?}{=} A_1 \cdot {y_{\mathcal{U}}}^c \pmod{p}$;
5. verifies $\alpha^{\mathsf{w}_2} \stackrel{?}{=} A_2 \cdot {\mathsf{H}_{\mathcal{U}}}^c \pmod{p}$;
6. computes the shared session key: $\mathsf{K} = hash(\mathsf{w}_2)$;
7. obtains a unique serial number Sn, and a random value $\mathsf{r}_{\mathcal{I}} \stackrel{R}{\leftarrow} \mathbb{Z}_p$;
8. computes $\mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n)} = hash^n(\mathsf{r}_{\mathcal{I}})$;

[11] http://sourceforge.net/apps/trac/silvertunnel/wiki/TorJavaOverview

9. composes $\kappa = (\mathsf{K}, \mathsf{r}_{\mathcal{I}})$ and signs it $\kappa^* = (\kappa, \mathsf{Sign}_{\mathcal{I}}(\kappa))$;
10. encrypts κ^* with a digital envelope which is decryptable by the TTP $\mathcal{T}$ and the provider $\mathcal{P}$ for possible future controversial situations during the ticket verification: $\delta_{\mathcal{T},\mathcal{P}} = \mathsf{pk}_{\mathcal{T},\mathcal{P}}(\kappa^*)$. This a mechanism that prevents $\mathcal{I}$ from forging $\mathsf{r}_{\mathcal{I}}$, because $\mathcal{T}$ can check that information and, demonstrate that $\mathcal{I}$ is the culprit;
11. fills out the ticket information T (Sn, Sv, $\mathsf{Pseu}_{\mathcal{U}}$, Tv, Ti, $\mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n)}$, $\mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n)}$, $\delta_{\mathcal{T},\mathcal{P}}$, etc.);
12. digitally signs the ticket T, and obtains the signed ticket, $\mathsf{Sign}_{\mathcal{I}}(\mathsf{T}) = \mathsf{sk}_{\mathcal{I}}(hash(\mathsf{T}))$, and $\mathsf{T}^* = (\mathsf{T}, \mathsf{Sign}_{\mathcal{I}}(\mathsf{T}))$;
13. sends T^* to the user $\mathcal{U}$.

receiveTicket $\mathcal{U}$ executes:

1. verifies the digital signature $\mathsf{Sign}_{\mathcal{I}}(\mathsf{T})$ of the ticket T using the issuer's certificate;
2. verifies that ticket T data and the performed request match;
3. verifies the ticket validity (T.Ti, T.Tv);
4. verifies $\mathsf{T.Pseu}_{\mathcal{U}}$;
5. stores $(\mathsf{T}^*, \mathsf{r}_{\mathcal{U}}, j = 0)$ in the device. We set up j to 0 because represents the occasions that the e-ticket has been used. The e-ticket will be consumed when $j = n$.

Ticket Verification When the user wants to use the service, she must verify the ticket in advance. For simplicity, we present the ticket verification with only one provider, so the service provider $\mathcal{P}$ never needs permanent communication with the ticket issuer. Nonetheless, the protocol can be extended for multiple providers. In that case, all the service providers should be connected to a central repository of spent tickets in order to control ticket overspending. The user only interacts with the service provider, but in controversial situations, she and/or the service provider could interact directly with the TTP through a resilient connection in order to preserve the security requirements of the protocol. If user misbehaved, her identity could be revoked, enabling to take further actions.

$\mathcal{U}$ sends the ticket T^*, and $\mathcal{P}$ checks it. If passed, $\mathcal{P}$ sends the commitment so that $\mathsf{r}_{\mathcal{I}}$ will be disclosed if $\mathcal{U}$ behaves correctly. Once the user sends the secret value $\mathsf{r}_{\mathcal{U}}$ encrypted through a shared key, then she receives the secret $\mathsf{r}_{\mathcal{I}}$ together with the receipt R^* from $\mathcal{P}$. The service provider $\mathcal{P}$ and the user $\mathcal{U}$ do the following steps:

showTicket $\mathcal{U}$ computes:

1. sends ticket $\mathsf{m}_1 = (\mathsf{T}^*, i)$ to $\mathcal{P}$. As a general case, we suppose that the service costs s of the n times that the e-ticket can be spent. So, the value i is computed as $i = j + s$;

verifyTicket $\mathcal{P}$ executes:

1. verifies the ticket signature, T.Sv, T.Ti, and T.Tv;
2. if the verifications fail, $\mathcal{P}$ omits m_1, and aborts the ticket verification;
3. else $\mathcal{P}$ looks for the ticket T^* in the database using T.Sn; and verifies that the ticket has not been spent by retriving the information related to the ticket $(j, \mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n-j)})$ in the provider's data base (if no information is found, then j is set to $j = 0$):
   (a) if $(i > j)$ then:
      i. computes $\mathsf{A}_{\mathcal{P},i} = PRNG(\mathsf{h}_{\mathsf{K}}) \oplus \mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n-i)}$, where $PRNG(\mathsf{h}_{\mathsf{K}})$ is a secure pseudorandom number generator and, $\mathsf{h}_{\mathsf{K}} = hash(\mathsf{K})$ is the seed. Note that K and $\mathsf{r}_{\mathcal{I}}$ are obtained from $\delta_{\mathcal{T},\mathcal{P}}$, then the provider is able to compute $\mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n-i)} = hash^{(n-i)}(\mathsf{r}_{\mathcal{I}})$;
      ii. encrypts $\mathsf{A}_{\mathcal{P},i}$ with the public key of the TTP $\mathcal{T}$: $\mathsf{pk}_{\mathcal{T}}(\mathsf{A}_{\mathcal{P},i})$;
      iii. stores $\mathsf{A}_{\mathcal{P},i}$ for future use;
      iv. assigns $\mathsf{V}_{\mathsf{succ}} = (\mathsf{T.Sn}, \mathsf{flag}_1, \tau_1, \mathsf{pk}_{\mathcal{T}}(\mathsf{A}_{\mathcal{P},i}), j)$, ($\tau_1$ is the verification timestamp). The flag flag_1 indicates that the ticket is valid and has not been spent yet. The signature is noted: $\mathsf{V}_{\mathsf{succ}}{}^* = (\mathsf{V}_{\mathsf{succ}}, \mathsf{sk}_{\mathcal{P}}(hash(\mathsf{V}_{\mathsf{succ}})))$;

v. sends $\mathsf{m}_2 = \mathsf{V}_{\mathsf{succ}}{}^*$ to $\mathcal{U}$;

(b) if $(i \leq j)$ then:

i. computes $\mathsf{h}_{(r_{\mathcal{U}},n-i)} = hash^{(j-i)}(\mathsf{h}_{(r_{\mathcal{U}},n-j)})$

ii. assigns $\mathsf{V}_{\mathsf{fail}} = (\mathsf{T.Sn}, \mathsf{h}_{(r_{\mathcal{U}},n-i)}, \mathsf{flag}_0, i, \tau_1)$. The flag_0 indicates that the ticket has been spent, i.e. is not valid. The signature is noted: $\mathsf{V}_{\mathsf{fail}}{}^* = (\mathsf{V}_{\mathsf{fail}}, \mathsf{sk}_{\mathcal{P}}(hash(\mathsf{V}_{\mathsf{fail}})))$;

iii. sends $\mathsf{m}_2 = \mathsf{V}_{\mathsf{fail}}{}^*$ to $\mathcal{U}$;

showProof $\mathcal{U}$ executes:

1. verifies $\mathcal{P}$'s signature;
2. if $\mathsf{V}_{\mathsf{succ}}{}^*$ or either $\mathsf{V}_{\mathsf{fail}}{}^*$ are not received, the *Claim* m_2 *Not Received* is called;
   (a) if $\mathsf{V}_{\mathsf{fail}}{}^*$ is received, $\mathcal{U}$ aborts the verification process. If the response is not correct, $\mathcal{U}$ can contact with the TTP to reconsider the situation by calling *Claim* m_2 *Not Received*;
   (b) if $\mathsf{m}_2 = \mathsf{V}_{\mathsf{succ}}{}^*$ is received, $\mathcal{U}$ has to verify the signature and data. If verifications are correct she continues the protocol. Otherwise, $\mathcal{U}$ can contact the TTP by calling *Claim* m_2 *Not Received*;
3. calculates $\mathsf{A}_{\mathcal{U},i} = PRNG(\mathsf{K}) \oplus \mathsf{h}_{(r_{\mathcal{U}},n-i)}$, using the shared value K as seed;
4. sends $\mathsf{m}_3 = (\mathsf{T.Sn}, \mathsf{A}_{\mathcal{U},i})$ to $\mathcal{P}$;

verifyProof $\mathcal{P}$ follows the next steps:

1. if $\mathsf{h}_{(r_{\mathcal{U}},n-i)}$ is not received, the *Claim* m_3 *Not Received* is called;
2. obtains $\mathsf{T.Sn}$, and computes $\mathsf{h}_{(r_{\mathcal{U}},n-i)} = \mathsf{A}_{\mathcal{U},i} \oplus PRNG(\mathsf{K})$;
3. verifies $\mathsf{h}_{(r_{\mathcal{U}},n-j)} \stackrel{?}{=} hash^{(i-j)}(\mathsf{h}_{(r_{\mathcal{U}},n-i)})$;
4. if $\mathsf{h}_{(r_{\mathcal{U}},n-i)}$ does not match, the *Claim* m_3 *Not Received* is called;
5. generates τ_2 and verifies it using the ticket expiry date $(\mathsf{T.Ti}, \mathsf{T.Tv})$ and the timestamp τ_1;
6. signs $\mathsf{A}_{\mathcal{P},i}$ approving then the validation with timestamp τ_2: $\mathsf{R} = (\mathsf{A}_{\mathcal{P},i}, \mathsf{T.Sn}, \tau_2)$, and $\mathsf{R}^* = (\mathsf{R}, \mathsf{sk}_{\mathcal{P}}(hash(\mathsf{R})))$;
7. stores and updates its data base: $[\mathsf{R}^*, (\mathsf{h}_{(r_{\mathcal{U}},n-j)} \blacktriangleleft \mathsf{h}_{(r_{\mathcal{U}},n-i)}), (j \blacktriangleleft i)]$ [12];
8. sends $\mathsf{m}_4 = \mathsf{R}^*$ to $\mathcal{U}$;

getValidationConfirmation $\mathcal{U}$ follows the next steps:

1. checks the signature of R^*;
2. computes $\mathsf{h}_{(r_{\mathcal{I}},n-i)} = \mathsf{A}_{\mathcal{P},i} \oplus PRNG(\mathsf{h}_{\mathsf{K}})$;
3. verifies $\mathsf{h}_{(r_{\mathcal{I}},n-j)} \stackrel{?}{=} hash^{(i-j)}(\mathsf{h}_{(r_{\mathcal{I}},n-i)})$;
4. if all verifications are correct, then stores and updates her data base $[\mathsf{R}^*, (\mathsf{h}_{(r_{\mathcal{U}},n-j)} \blacktriangleleft \mathsf{h}_{(r_{\mathcal{U}},n-i)}), (j \blacktriangleleft i)]$; or else calls *Claim* m_4 *Not Received* to the TTP.

The *Ticket Verification* protocol is a fair exchange protocol with the existence of an offline TTP [37] between the user and the provider of the service (a valid e-ticket is given in exchange for the permission to use the service). This enables dispute resolution protocols in case of incorrect behaviour of the actors so as to preserve the security of the system. In case of dispute, they can contact the TTP following these protocols:

Claim m_2 Not Received This protocol can be executed if $\mathcal{U}$ sends m_1 and says that she has not received $\mathsf{m}_2 = \mathsf{V}_{\mathsf{succ}}{}^*$ from $\mathcal{P}$.

Claim User $\mathcal{U}$ executes:

1. sends the ticket $\mathsf{m}_1 = (\mathsf{T}^*, i)$ to the TTP $\mathcal{T}$;

Response TTP $\mathcal{T}$ follows the next steps:

1. checks the information, signature and timestamp;

[12] The notation $a \blacktriangleleft b$ is used to describe a database update operation where the value represented by a is replaced by b

2. if the verification is correct, generates $(\mathsf{T.Sn}, \tau_3)$; then
3. signs the information $\mathsf{m}_5 = ((\mathsf{T.Sn}, \tau_3), \mathsf{sk}_{\mathcal{T}}(hash((\mathsf{T.Sn}, \tau_3)))$; and
4. sends m_5 to both $\mathcal{U}$ and $\mathcal{P}$. This entails acceptance of $\mathcal{U}$'s sent information and then $\mathcal{P}$ has the responsibility to unblock and send a correct m_2 to continue with the verification phase at sub-phase *verifyTicket*. After that, if the service could not be finally guaranteed, $\mathcal{U}$ could demonstrate to a third party (by showing m_5) that $\mathcal{U}$ behaved correctly and $\mathcal{P}$ was the responsible of the denial of service;

verifyTicketWithTTP Service provider $\mathcal{P}$ executes:

1. executes *verifyTicket* normally;
2. sends m_2 to both $\mathcal{T}$ and $\mathcal{U}$, and continues the *Ticket Verification* steps at point *showProof*. The TTP has to store m_2 and m_5 because the user can go to an external dispute resolution system (if m_2 is still wrong) to solve the problem. In this case, the TTP will be able to provide these evidences;

Claim m_3 Not Received This protocol can be executed if $\mathcal{P}$ sends m_2 and says that has not received $\mathsf{m}_3 = \mathsf{A}_{\mathcal{U},i}$ (with a correct $\mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n-i)}$ inside) from $\mathcal{U}$.

Claim Provider $\mathcal{P}$ executes:

1. blocks the ticket $\mathsf{T.Sn}$ till the reception of m_3 from $\mathcal{U}$ or m_5 by $\mathcal{T}$;
2. another $\mathsf{T.Sn}'$ received from the same connection could not be accepted and $\mathsf{m}_2 = \mathsf{V}_{\mathsf{succ}}{}^*$ would be repeatedly sent in order to unblock the ticket identified by $\mathsf{T.Sn}$.

Claim m_4 Not Received This protocol can be executed if $\mathcal{U}$ sends m_3 and says that has not received $\mathsf{m}_4 = \mathsf{R}^*$ (with the contained $\mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n-i)}$) from $\mathcal{P}$.

Claim User $\mathcal{U}$ follows the next steps:

1. sends to the TTP $\mathcal{T}$: $(\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3) = (\mathsf{T}^*, \mathsf{V}_{\mathsf{succ}}{}^*, (\mathsf{T.Sn}, \mathsf{A}_{\mathcal{U},i}))$;

Response TTP $\mathcal{T}$ executes:

1. checks the entire information; if verification fails, aborts the claim;
2. computes $\mathsf{A}_{\mathcal{P},i} = PRNG(\mathsf{h}_{\mathsf{K}}) \oplus \mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n-i)}$ using K and $\mathsf{r}_{\mathcal{I}}$. Note that K and $\mathsf{r}_{\mathcal{I}}$ can be obtained by decrypting $\delta_{\mathcal{T},\mathcal{P}}$ and then $\mathcal{P}$ can computes $\mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n-i)} = hash^{(n-i)}(\mathsf{r}_{\mathcal{I}})$;
3. checks that $\mathsf{A}_{\mathcal{P},i}$ (from the previous step) $\stackrel{?}{=} \mathsf{m}_2.\mathsf{V}_{\mathsf{succ}}.\mathsf{A}_{\mathcal{P},i}$;
4. verifies that $\mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n-i)}$ (computed at step 2) matches with $\mathsf{T.h}_{(\mathsf{r}_{\mathcal{I}},n)}$;
5. checks that $\mathsf{m}_1.i > \mathsf{m}_2.\mathsf{V}_{\mathsf{succ}}.j$
6. computes $\mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n-i)} = \mathsf{A}_{\mathcal{U},i} \oplus PRNG(\mathsf{K})$ and checks that $hash^i(\mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n-i)}) \stackrel{?}{=} \mathsf{T.h}_{(\mathsf{r}_{\mathcal{U}},n)}$
7. if everything is successful, then generates $(\mathsf{T.Sn}, \mathsf{A}_{\mathcal{P},i}, \mathsf{A}_{\mathcal{U},i}, \tau_4)$; otherwise, publishes which entity behaved corruptly in accordance with the above verifications;
8. signs the information $\mathsf{m}_6 = ((\mathsf{T.Sn}, \mathsf{A}_{\mathcal{P},i}, \mathsf{A}_{\mathcal{U},i}, \tau_4), \mathsf{sk}_{\mathcal{T}}\ (hash((\mathsf{T.Sn}, \mathsf{A}_{\mathcal{P},i}, \mathsf{A}_{\mathcal{U},i}, \tau_4)))$;
9. sends m_6 to $\mathcal{U}$.

Outside the ambit of the telecommunications, m_6 can be used as an evidence in case of a user demand for the right to use the service in an external dispute resolution system.

C.2.2 Multiple Providers

The described proposal considers that only one provider is able to give a certain service, enabling then offline verification. Nevertheless, this scenario could be extended to the existence of multiple providers that give a certain service and accept the same ticket in different places, but guaranteeing the control of ticket overspending through

online verification between all the providers. In this extended scenario, $\mathsf{sk}_{\mathcal{P}}$ would be shared between the providers, enabling the access to K and $\mathsf{r}_{\mathcal{I}}$.

There would be also special care to the distribution and control of used tickets (controlled by the existence of $\mathsf{r}_{\mathcal{U}}$ in the database for that ticket). There should be a central database where all the providers could store all the used tickets, and then the verification would be online by imperative. Expired tickets could be removed from the database for storage efficiency, and, moreover, only ticket serial number could be stored in the database despite storing all the ticket information.

C.3 Security and privacy of the system

Proposition 1 *The proposed e-ticketing system preserves authenticity, non-repudiation, integrity and the expiry date of the e-ticket.*

Claim I. *t is computationally unfeasible to make a new fraudulent e-ticket.*

Security Argument. *A valid e-ticket has the form* $\mathsf{T}^* = (\mathsf{T}, \mathsf{Sign}_{\mathcal{I}}(\mathsf{T}))$. *Then, the first step that the provider* $\mathcal{P}$ *does when an e-ticket is received is the verification of the signature. The* Ticket Verification *protocol will continue only if this verification ends correctly; otherwise,* $\mathcal{P}$ *refuses* $\mathcal{U}$*'s request. Thus, making a new fraudulent valid e-ticket would be equivalent to break the signature scheme and that would be computationally unfeasible as we have supposed that the issuer* $\mathcal{I}$ *uses a secure signature scheme.*

Claim T. *he issuer can not deny the emission of a valid e-ticket.*
Security Argument. *A valid e-ticket has* $\mathcal{I}$*'s signature and the signature scheme used is secure. Consequently, the identity of the issuer is associated to the ticket; the signature is a non-repudiation evidence of origin.*

Claim T. *he content of the e-ticket can not be modified.*
Security Argument. *Suppose that someone modifies the content of the ticket, then a new* $\mathcal{I}$*'s signature has to be generated over the modified content; otherwise, the e-ticket will not be valid. Again, if it is computationally unfeasible to forge the* $\mathcal{I}$*'s signature, it is unfeasible to modify the content of the e-ticket.*

Claim T. *he e-ticket will not be longer valid after the ticket validity time* $\mathsf{T.Tv}$.
Security Argument. *The provider* $\mathcal{P}$ *receives the e-ticket from the user at the* Ticket Verification *protocol before allowing access to the service. First of all* $\mathcal{P}$ *checks the correctness of the e-ticket (obviously that includes the verification of* $\mathsf{T.Tv}$*). If the verification is not correct* $\mathcal{P}$ *stops the protocol and the user will have no access to the service. Also, according the* Claim 3*, the user can not tamper* $\mathsf{T.Tv}$.

Result of Proposition 1 *According to the definitions given at Section C.2 and the Claims 1, 2, 3 and 4 we can assure that the protocol achieves the properties specified at Proposition 1.*

Proposition 2 *The e-ticketing system described in Section C.2 is anonymous. The service offered is revocable anonymous.*

Claim A. *n e-ticket is anonymous.*
Security Argument. *A valid e-ticket has the following information* $\mathsf{T} = (\mathsf{Sn}, \mathsf{Sv}, \mathsf{Pseu}_{\mathcal{U}}, \mathsf{Tv}, \mathsf{Ti}, \mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n)}, \mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n)}, \delta_{\mathcal{T},\mathcal{P}}, \ldots)$*. The information related to the user's identity is solely* $\mathsf{Pseu}_{\mathcal{U}} = (y_{\mathcal{U}}, \mathsf{sk}_{\mathcal{T}}(hash(y_{\mathcal{U}})))$*, where* $y_{\mathcal{U}} = \alpha^{x_{\mathcal{U}}} \pmod{p}$*. The user's identity is* $x_{\mathcal{U}}$*, thus an enemy has to solve the problem of computing the discrete logarithm to know the identity of the user. Currently no efficient algorithms are known to compute that.*

Claim T. *he purchase of an e-ticket is anonymous.*
Security Argument. *As the protocol in Section C.2.1 specifies, the channel between* $\mathcal{U}$ *and* $\mathcal{I}$ *of the ticket is anonymous. The protocol uses a Schnorr's ZKP to provide the user identity to the* $\mathcal{I}$*, so that the issuer can be sure that the connected user who wants to buy the ticket is the right holder of the pseudonym* $\mathsf{Pseu}_{\mathcal{U}}$ *without disclosing her real identity. Thus, the user does not need to reveal her identity to buy an e-ticket.*

Claim A. *fake user cannot buy an e-ticket impersonating other user.*
Security Argument. *In order to buy a ticket, the user has to perform a Schnorr's ZKP to prove knowledge of the identity to the issuer without revealing it. The user has to compute* w_1 *such as* $\alpha^{\mathsf{w}_1} \stackrel{?}{=} A_1 \cdot {y_{\mathcal{U}}}^c \pmod{p}$*. As far as any user preserves the privacy of her identity* $x_{\mathcal{U}}$ *(which links to* $\mathsf{Cert}_{\mathcal{U}}$ *through the cooperation of* $\mathcal{T}$*), anyone else will not be able to compute such* w_1*. In this case, user can only be accused through* $x_{\mathcal{U}}$ *of ticket overspending (supposing that the user keeps* $x_{\mathcal{U}}$ *secretly), because she solely has the information to perform the* Ticket Verification *protocol. Thus, the e-ticketing system also preserves the exculpability property.the ticket related to the exculpability proof. The identity of the user could not be used to falsely accuse* $\mathcal{U}$*.*

Result of Proposition 2 *According to the definitions given at Section C.2 and the Claims 5, 6 and 7 we can assert the Proposition 2. The e-ticketing system is anonymous and this anonymity could be revocable in case of a user's fraudulent action. The pseudonym manager* $\mathcal{T}$ *knows the correspondence between* $x_{\mathcal{U}}$ *and* $y_{\mathcal{U}}$ *(see Algorithm: 'Pseudonym Renewal'). Therefore,* $\mathcal{T}$ *could reveal the association between* $x_{\mathcal{U}}$ *and* $y_{\mathcal{U}}$ *due to law enforcement (e.g. a judge could request the user's identity to* $\mathcal{T}$*).*

Proposition 3 *The protocol satisfies the property of exculpability.*

Claim T. *he service provider can not falsely accuse the user of ticket overspending.*
Security Argument. *When the service provider* $\mathcal{P}$ *receives the message* m_1 *in step 1 of the* Ticket Verification *protocol* (showTicket)*,* $\mathcal{P}$ *looks for the ticket that matches with the received serial number in its database. If the ticket had been already spent, the service provider will find the overspending proof* $\mathsf{r}_{\mathcal{U}}$ *together with the ticket. The service provider has to show this element to accuse the user of overspending. If the user had not validated the ticket before, then the service provider does not have the element (*$\mathcal{U}$ *will send it in step 3:* showProof*), as the hash invertible function is believed to be computationally infeasible, and also there can not exist collisions in this hash function, so* $\mathcal{P}$ *can not falsely accuse the user of overspending. If the service provider, even not being able to prove the overspending, decides to deny the service to the user, the user can contact the TTP in order to solve the situation through* Claim m_2 Not Received*.*

Claim T. *he user* $\mathcal{U}$ *is able to prove that she has already validated the ticket.*
Security Argument. *If a user* $\mathcal{U}$ *executes successfully the* Ticket Verification *protocol,*

$\mathcal{U}$ will obtain the exculpability proof $\mathsf{r}_{\mathcal{I}}$. She can use this proof to demonstrate that the ticket has been validated. If the Ticket Verification *protocol is stopped and $\mathcal{U}$ does not obtain the exculpability proof after the revelation of $\mathsf{r}_{\mathcal{U}}$, she can execute* Claim m_4 Not Received*. This way $\mathcal{U}$ would obtain an alternative exculpability proof from the TTP.*

Result of Proposition 3 *According to the definitions given at Section C.2 and the Claims 8 and 9 we can assure that the protocol achieves the property specified at the Proposition 3. The ticket verification process is a fair exchange: any part can obtain the exculpability proof of the other part without revealing its own proof.*

Proposition 4 *The tickets issued by the protocol described in Section C.2.1 can be preset to be reusable tickets, both for a limited number of validations or a limited period of time.*

Claim T. *he protocol allows the creation of N-usable tickets maintaining the security properties of the non reusable tickets, including exculpability. Security Argument.*

During the first execution of the *Ticket verification*, $\mathcal{U}$ uses the last element of the chain of proofs $\mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n)}$ and receives in exchange an element containing the last element of the chain of issuer proofs $\mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n)}$. Due to the properties of hash functions, $\mathcal{U}$ cannot generate $\mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n-i)}$ and $\mathcal{P}$ cannot generate $\mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n-i)}$. The successive validations will use the remaining elements of the chain in reverse order.

Claim T. *he protocol allows the creation of period-usable tickets maintaining the security properties of the non reusable tickets, including exculpability. Security Argument.* In this case the concept of overspending is not applicable. The user will obtain a validation proof each time he executes the *Ticket Verification* protocol obtaining an exculpability proof provided the time of the verification attempt is less than the limit of the validity period.

Result of Proposition 4 *According to the Claims 10 and 11 we can assert the Proposition 4. The protocol is flexible enough to be used with all kinds of services, with independence of its reusability requirements.*

Proposition 5 *The protocol avoids overspending with minimum requirements of persistent connections with a centralized system.*

Claim T. *he protocol avoids overspending.*

Security Argument. *If a user tries to overspend a ticket, then initiates the verification another time after the first validation of the ticket. The user sends T^* to $\mathcal{P}$* (showTicket) *and waits for a response. $\mathcal{P}$ executes* (verifyTicket) *and looks for the ticket T^* in its database. Overspending can be prevented; the provider can prove it while $\mathcal{P}$ can not accuse the user of overspending attempts if $\mathcal{P}$ can not provide $\mathsf{r}_{\mathcal{U}}$.*

Claim I. *f the ticket can only be validated by one provider then the verification is totally offline.*

Security Argument. *The provider $\mathcal{P}$ maintains a database with the serial numbers of the e-tickets that have been already validated (together with their exculpability proofs) until their expiry date. With the contents of this database the provider has enough information to decide if $\mathcal{P}$ accepts and validates a new ticket because $\mathcal{P}$ can check both*

the issuer's signature and the fact that the e-ticket has not been spent before. So the provider does not need to contact to any party during the validation of an e-ticket.

CLAIM I. *f the ticket can be validated with several providers then the providers must be connected and share a database of spent tickets.*
Security Argument. *The set of providers maintains a shared database with the serial numbers of the e-tickets that have been already validated (together with their exculpability proofs) until their expiry date. The contents of this database are used by the providers to decide if they accept and validate a new ticket. So the provider does not need connection to the issuer during the verification of an e-ticket, but the set of providers must have a shared database instead.*

Result of Proposition 5 *According to the definitions given at Section C.2 and the Claims 10, 11 and 12 we can assure that the protocol achieves Proposition 5. The issuer is offline during the verification phase and the providers contact among them only in some kind of services. In all cases, the protocol prevents ticket overspending.*

C.4 Implementation details and experimental results

There are several important factors to consider when we design an e-ticketing system that should be usable in practice. The response time is one of them. Thus, we have implemented our protocol and we present some results regarding its time performance.

In Section C.4.1, we describe the developed components, the development environment and the hardware that we have used. Next, the testing methodology is described in Section C.4.2, i.e. the system can be configured using different key lengths. We can assume that we would obtain more security with larger keys but the computational cost will be higher. We want to study how the key length influence the computational cost. Next, we present the results obtained in Section C.4.3.

C.4.1 e-ticketing system development and configuration details

As introduced in Section C.2, our system comprises three main phases: pseudonym renewal, ticket purchase and ticket verification, and four actors: the user, the service provider, the ticket issuer and the pseudonym manager. So that, the system implementation requires four components: one for each entity in the system. Nonetheless, we have grouped in one server the service provider, the ticket issuer and the pseudonym manager for practical reasons, see Figure 7. The server takes the role of different servers in a PC. The server component has been developed with the Java programming language (Java 2 Standard Edition), allowing portability in a great number of platforms.

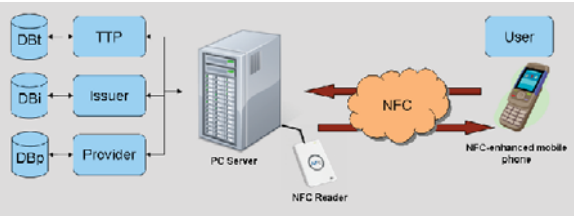


Figure 7: Architecture of the testing environment

The user interacts with the other actors (the service provider, the ticket issuer and the pseudonym manager) by means of a mobile phone, so that the user component (client) should be executed in a mobile phone. The Figure 8 shows the protocols implemented in the client component. Given that a great number of mobile phones can execute Java applications, we have developed the client in Java 2 Micro Edition (J2ME). The mobile phone used has been a Nokia 6212 Classic with an embedded API for NFC communication.

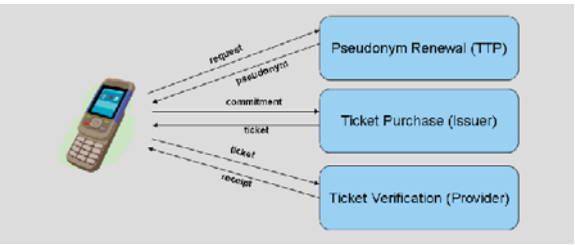


Figure 8: Protocols of the client component

The communication between server and client is performed via Near Field Communication (NFCIP-1, ISO18092). Nowadays, there are few mobile phones with NFC technology. Nonetheless, in a near future the main smart-phones platforms will include the NFC technology. Nokia has announced that from next year (2011) every Nokia smartphone will have NFC [38]. Android Gingerbread 2.3 will support near field communications to read RFID tags as well as communicate with other phones, payment systems and other possible applications [39], and Apple is testing an iPhone with NFC chips [40]. The server uses an Arygon NFC Reader (ADRA-USB) in order to connect with the mobile phone. The equipment of the entire scheme is detailed in Table 3.

It should be taken into consideration that the mobile phone acts as the initiator of the transactions, and the server is the target, i.e. the server is waiting for $\mathcal{U}$'s requests.

C.4.2 Testing methodology

The e-ticketing system can be configured with the key length parameter l. This parameter l refers to the key size (in bits) of the RSA cryptosystem used in the protocol, as

well as the number of bits of the generated prime numbers for the generation of the $\mathbb{Z}_q$ and $\mathbb{Z}_p$. The larger the parameter is, the harder is the cryptosystem to be broken, i.e. we have a system more secure. On the other hand, the time consumption is also increased, and has to be evaluated.

We have run the protocols with different key sizes of 512, 1024 and 2048 bits, respectively. The results are studied in Section C.4.3. Regarding the length of the keys l, at the present time a size of $l = 1024$ bits is considered computationally safe [41]. According to that, we have tested our scheme with a smaller length ($l = 512$ bits) and a larger one ($l = 2048$ bits). In this way, we can examine how the key length influences the system's performance.

Finally, we have studied the partial times of each protocol (pseudonym renewal, ticket purchase and ticket verification, see Sections C.4.4, C.4.5 and C.4.6 respectively), in order to identify the most costly parts of every protocol.

We have executed several test for every key length and protocol, so that the times shown in the following sections are the average of these times.

C.4.3 Global time cost results

Figure 9 shows the average time (in ms) required to complete each transaction (*Pseudonym Renewal*, *Ticket Purchase* and *Ticket Verification* phases) taking into account the interaction with the other entities. These results are given depending on the used key length l (in bits) with its values 512, 1024 and 2048, respectively. We focus specially on the *Ticket Verification* phase, where the delay time has to be strongly reduced if we assume a mass-transit scenario. This delay varies from 1 to 2 seconds depending on the key length l parameter, what makes the proposal definitely practical. In general, all the transactions are considered practical in 1024 bits (cost lower than 2s), and they become increased in 2048 bits. We detail the times of each phase by considering the costs of all their subphases in Sections C.4.4, C.4.5 and C.4.6 in order to show the most costly operations in terms of delay times.

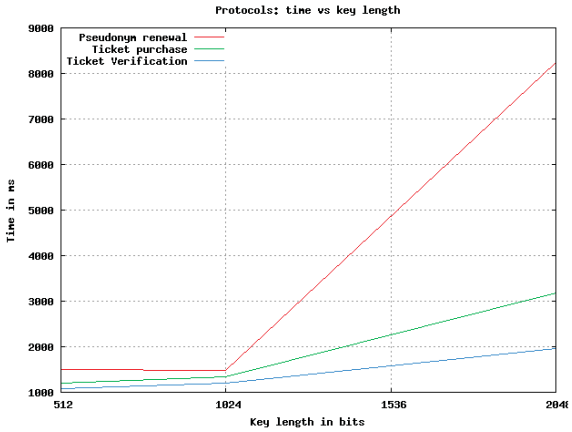


Figure 9: Computational cost of every protocol using several key lengths

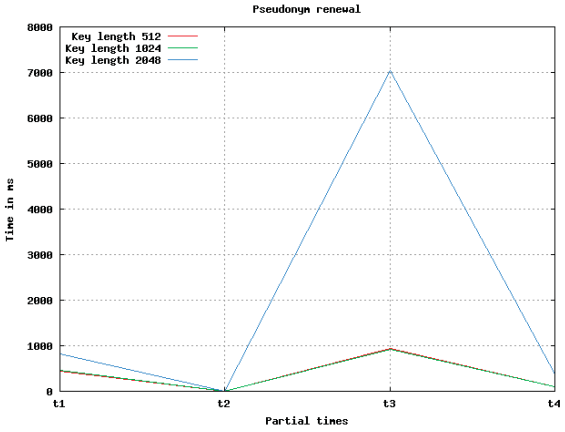


Figure 10: Partial times of the pseudonym renewal (see Table 4 for t_i details)

C.4.4 Detailed time cost of the pseudonym renewal

Figure 10 shows the partial time intervals of the *Pseudonym Renewal* phase. As expected, the decryption of the signed pseudonym (t_3) is the most costly operation in this phase, and increases obviously depending on the key length l parameter. This operation is performed by the user in the mobile phone. There are not great remarks in the other operations, as precomputation of the not-interactive values helps to reduce the global transaction times. pseudonym, and t_4 is the signature verification of this pseudonym.

C.4.5 Detailed time cost of the ticket purchase

Figure 11 shows the partial time intervals of the *Ticket Purchase* phase. The main costs remain on the computation and transmission of the Schnorr's ZKP (t_3 and t_4), as well as the communication cost of the first commitment (t_1), specially all of them with 2048 bits. The verification of the ticket signature (t_7) varies from 100 to 400 ms. Once again, some values have been precomputed to reduce the protocol execution.

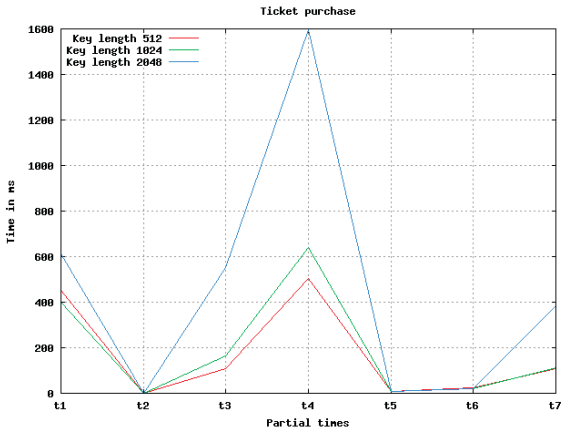


Figure 11: Partial times of the ticket purchase (see Table 5 for t_i details)

C.4.6 Detailed time cost of the ticket verification

Figure 12 shows the partial time intervals of the *Ticket Verification* phase. The most remarkable costs remain on the connection and sending of the ticket (t_1), depending on the amount of data with its key length (parameters and signature), followed by the signature verification of the response (t_3), the sending of the symmetric encryption of the parameter $\mathsf{r}_{\mathcal{U}}$ (t_5), and finally the verification of the receipt (t_7). Other times such as the reception of the response (t_2), the computation of the symmetric encryption of $\mathsf{r}_{\mathcal{U}}$ (t_4), the reception of the receipt (t_6) and the computation of the symmetric decryption of $\mathsf{r}_{\mathcal{I}}$ (t_8) have become not costly operations.

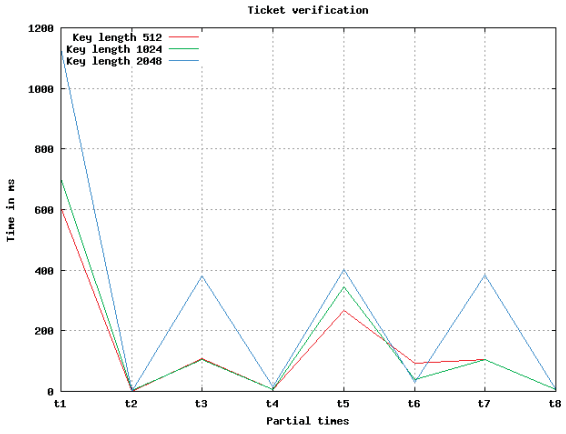


Figure 12: Partial times of the ticket verification (see Table 6 for t_i details)

Independently from the key length l parameter, there is a variation in the communication times depending on the steps of the protocol, as the server and the client have to synchronize their protocol steps in order to exchange their information.

D Secure Automatic Fare Collection System for Time-based or Distance-based Services with Revocable Anonymity for Users

The incorporation of Information and Communication Technologies (ICT) in Automatic Fare Collection (AFC) systems allows to reduce costs and it improves the control of the infrastructures; some examples could be the real-time traffic density monitorization and the management strategy of infrastructures depending on the passenger flows.

AFC systems are designed for massive-density public transports. Instead of setting the user's destination or a parking place, the fare can be calculated on time-based and/or distance-based system. Benefits of using such systems include the elimination of cash and vending machines, customer convenience, faster travel and reduced the accounting and back-office costs. However, to achieve these that it's necessary the design of efficient and secure AFC systems. Thus, a secure management of users' *check-in* and *check-out* of the system is needed, as they pay according to this use.

If the system identifies the users, knowing their entrance and exit points, i.e. where they go and when, the system (a company) can trace their movements. Using these movements it can create a profile of each user. This is a serious privacy threat, i.e. it violates the users' privacy. For this reason, these AFC systems have to preserve the users' privacy in order to prevent tracking and profiling. Nonetheless, if the system offers non-revocable privacy, some criminals, for instance terrorists, can use the transport system to scape from the policy. Thus, we have two opposed requirements. In order to fulfill both requirements, the system could revoke the user's anonymity with a court order. Another consideration is the high number of users in an AFC system, so that, the system must be very fast in the operations of entrance and exit.

Our work offers a secure management for AFC systems, with strong privacy for honest users. However, if the user acts dishonestly, her identity can be disclosed in order to take legal actions, i.e. the system offers revocable anonymity. Moreover, the users do not need to obtain a new credential every time that they use the AFC system. This feature improves the system usability, because in the previous AFC systems new credential information is needed for every new journey.

D.1 State-of-the-Art

We have performed an analysis of AFC proposals that consider anonymity for users, offering revocable anonymity [42, 43, 31, 44, 45, 46].

In the majority of these schemes, the provider can link different journeys from the same user [42, 43, 44, 45, 46]. In a linkable system, the disclosure of the identity of the user in a journey leads to the disclosure of all the journeys of the same user (weak anonymity). So that, the provider knows where they go, when and the time

of the journeys. The knowledge of users' behavior allows the creation of the users' profiles. The profiles are useful for the provider because they can be used to improve the transportation system or to define a commercial product specifically for one profile. Nonetheless, the creation of users' profiles is a serious violation of the privacy, i.e. the AFC system must avoid the tracking of the users.

In [31] the provider can not trace these journeys, but then a new credential is needed for every journey, what means that there is an important extra cost in these mass-transport systems, where the entrances and exits of the system have to be as quick as possible. The credential renewal requires a more complex provider structure, i.e. costly, because it must manage a high number of credentials.

Related to the devices used in the proposals, the latest trends go in the direction to use mobile devices (e.g. mobile phones, PDAs, smart phones, etc.) [42, 31, 45, 46] for these systems, instead of smart cards [43, 31, 44]. Thus, we can say that the mobile device is a user's requirement in the AFC systems.

In Table 7 we classify the analyzed proposals depending on the level of anonymity guaranteed for users, the availability to trace different journeys of a same user, and the devices used in that systems.

Our system offers revocable anonymity and untraceability for users. Moreover, this system has been designed in order to use the personal mobile devices of the users in the system, and, in addition, users do not need to obtain a new credential every time they perform a journey, differently than in [31], where the credential renewal means an important extra cost.

The fare to be paid is calculated in each service using the relevant parameters for the Collection. A classification can be made base on these parameters. The services where the parameter to be priced is the time will be called Time-Based Fare Collection systems while the services that charge the distance covered are called Distance-Based Fare Collection systems.

D.1.1 Time-Based Systems

The most common Time-Based Fares are the daily, weekly or monthly passes used by citizens to employ public facilities such as public swimming pools and other sports facilities. The concept can be extended to the public transport, creating a time-based individual ticket that allows the passenger to make use of a transit system and make free transfers for a set amount of time. However, in this paper, we will refer to the Time-Based Fares Systems when the amount to be paid by the customer depends on period of time that the service has been used. Thus, in this case, a proper timestamp has to be generated when the user get in the system. The difference between the present time and the initial timestamp will determine the fare to be paid to the provider of the service. Although the implementation of Time-Based AFC systems is proven challenging in transport systems, there are some experiences worldwide in this area. For instance, the DIMTS (Delhi Integrated Multi-Modal Transport System) from India has an AFC system, which tries to accommodate various types of existing passes/tickets and it is expected to cater new time-based fares (peak and off-peak differential fare). Also Hyundai Information Technology in Korea has developed AFC systems. We can find more examples of AFC systems in public transport with Time-Based AFC options

in Sydney (Australia) or San Diego (USA).

D.1.2 Distance-Based Systems

Distance-based AFC systems are more common tan Time-based systems. Some examples of Distance-Based Collection Systems are the tolls and public transport services as the subway or the bus. In distance-based AFC systems the fare to be paid depends on the distance between the entrance point and the exit point. These points are associated with the providers' stations. The function that calculates the fare to be paid can use other entrance parameters as the kind of vehicle in a toll or the day or the hour a public transport is used. These services can be more complex than those priced by time. While time always moves forward, the distance between two points can be covered in two opposite directions. This fact can be the base of some confabulated attacks that will be described in §D.6.3.

D.1.3 Contribution

According to the experiences and pilot test in many cities, the implementation of AFC system in the public transport has proven challenging. However, as new technologies emerge, such as mobile phones, they will help to provide suitable AFC systems.

In our previous work [47], we proposed an automatic fare collection system for distance-based systems, where users were not able to change the direction of the movement without exiting the service, i.e., they must check out according to their direction. This required that system exits be separated by direction.

In our proposal, we extend our work in order to allow time-based systems and distance-based systems that are not separated by direction. Thus, we present two protocols, one for based-time and other for distance-based systems. The protocols use a group signature (BBS) scheme in order to verify that a user is a correct member of a certain group of users. The group signature scheme used is described in section D.2, and we present a method that allows to link two group signatures. Next, a requirements' study of the security for time-based and distance-based protocols is conducted in section D.3. Note that [47] does not include the requirements for the above extensions. The time-based protocol is described in section D.5 and the distance-based in section D.6. The section D.7 contains the security analysis of both protocols. The protocols' implementation is depicted in section D.8. The protocols have been implemented in the Android mobile platform and its performance has been evaluated in two Android smartphones.

D.2 Background

We use the short group signature (BBS) scheme [48] in order to verify that a user is a correct member of a certain group of users. For that reason, we present here their main definitions. Note that the notation in this section is specific for the explanation of the used definitions; in our protocol definition, only the procedures ($KeyGen_G, Sign_G, Verify_G$, $Open_G, SignLinkable_G, VerifyLinkable_G$) will be further called with their parameters, not their internal details.

Consider bilinear groups G_1 and G_2 with respective generators g_1 and g_2. Suppose further that the SDH assumption holds on (G_1, G_2), and the Linear assumption holds on G_1. The scheme uses a bilinear map $e : G_1 \times G_2 \rightarrow G_T$ and a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.
The public values are $g_1, u, v, h \in G_1$ and $g_2, w \in G_2$. Here $w = g_2^\gamma$ for some secret $\gamma \in \mathbb{Z}_p$.

- $KeyGen_G(n)$. This algorithm takes as input a parameter n, which is the number of members of the group. Then select $h \xleftarrow{R} G_1 \backslash \{1_{G_1}\}$ and $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$, and set $u, v \in G_1$ such that $u^{\xi_1} = v^{\xi_2} = h$. Select $\gamma \xleftarrow{R} \mathbb{Z}_p^*$ and set $w = g_2^\gamma$. Generate for each user $\mathcal{U}_i$, $1 \leq i \leq n$, an SDH tuple (A_i, x_i) by performing: select $x_i \xleftarrow{R} \mathbb{Z}_p^*$ and set $A_i \leftarrow g_1^{1/(\gamma + x_i)}$. The parameter γ is then the private master key of the group key issuer.
- $Sign_G(gpk, gsk[i], M)$. Given a group public key $gpk = (g_1, g_2, h, u, v, w)$, a private user's key $gsk[i] = (A_i, x_i)$ and a message $M \in \{0,1\}^*$, compute and output a signature of knowledge $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$.
  1. select $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$ and compute the linear encryption of A: $(T_1, T_2, T_3) \leftarrow (u^\alpha, v^\beta, Ah^{\alpha+\beta})$ together with the helper values $\delta_1 \leftarrow x\alpha$ and $\delta_2 \leftarrow x\beta$;
  2. select $r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2} \xleftarrow{R} \mathbb{Z}_p$ and compute the values:
  $$R_1 \leftarrow u^{r_\alpha}, R_2 \leftarrow v^{r_\beta}$$
  $$R_3 \leftarrow e(T_3, g_2)^{r_x} \cdot e(h, w)^{-r_\alpha - r_\beta} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}$$
  $$R_4 \leftarrow T_1^{r_x} \cdot u^{-r_{\delta_1}}, R_5 \leftarrow T_2^{r_x} \cdot v^{-r_{\delta_2}}$$
  3. self-compute the challenge:
  $$c \leftarrow H(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5)$$
  4. compute the values $s_\alpha \leftarrow r_\alpha + c\alpha, s_\beta \leftarrow r_\beta + c\beta, s_x \leftarrow r_x + cx, s_{\delta_1} \leftarrow r_{\delta_1} + c\delta_1, s_{\delta_2} \leftarrow r_{\delta_2} + c\delta_2$
  5. output $\sigma \leftarrow (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$.
- $Verify_G(gpk, M, \sigma)$. Given a group public key $gpk = (g_1, g_2, h, u, v, w)$, a message M and a group signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$, verify that σ is a valid signature of knowledge.
  1. re-derive R_1, R_2, R_3, R_4, R_5:
  $$\tilde{R_1} \leftarrow u^{s_\alpha} / T_1^c, \tilde{R_2} \leftarrow v^{s_\beta} / T_2^c$$
  $$\tilde{R_3} \leftarrow e(T_3, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha - s_\beta} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot (e(T_3, w)/e(g_1, g_2))^c$$
  $$\tilde{R_4} \leftarrow T_1^{s_x} / u^{s_{\delta_1}}, \tilde{R_5} \leftarrow T_2^{s_x} / v^{s_{\delta_2}}$$
  2. checks that $c \stackrel{?}{=} H(M, T_1, T_2, T_3, \tilde{R_1}, \tilde{R_2}, \tilde{R_3}, \tilde{R_4}, \tilde{R_5})$.

- $Open_G(gpk, gmsk, M, \sigma)$. This algorithm is used in order to trace a signature to a concrete signer inside the group. It is only available for the group manager, as he is the holder of the *gmsk* master key, and knows all the pairs (A_i, x_i). Given a group public key $gpk = (g_1, g_2, h, u, v, w)$, the group master private key $gmsk = (\xi_1, \xi_2)$, together with a message M and a signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$, it proceeds as follows. First, recover the user's A by performing $A \leftarrow T_3/(T_1^{\xi_1} \cdot T_2^{\xi_2})$. If the group manager is given the elements $\{A_i\}$ of the user's private keys, he can look up the user index corresponding to the identity A recovered from the signature.

D.2.1 Linkability between signatures - Procedure $SignLinkable_G$

We define a new linkable signing procedure called $SignLinkable_G(gpk, gsk[i], M)$ to be used in the protocol. Given a group public key gpk, a private user's key $gsk[i]$ and a message M, compute and output a signature of knowledge σ. In order to use this procedure correctly, we recommend to use it in the protocol as follows:

- First use: standard $Sign_G(gpk, gsk[i], M)$:
  - generate a linear encryption of A:
    $(T_1, T_2, T_3) \leftarrow (u^\alpha, v^\beta, Ah^{\alpha+\beta})$ for $\alpha, \beta \overset{R}{\leftarrow} \mathbb{Z}_p$;
  - given a message M, sign the message and output a signature
    $\sigma \leftarrow (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ where
    $c \leftarrow H(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5) \in \mathbb{Z}_p$;
- Further uses: $SignLinkable_G(gpk, gsk[i], M)$:
  - use the same pair (α, β) producing the same linear encryption of A than in the first time:
    $(T_1, T_2, T_3) = (u^\alpha, v^\beta, Ah^{\alpha+\beta})$;
  - given a message M', sign the message and output a signature
    $\sigma' \leftarrow (T_1, T_2, T_3, c', s'_\alpha, s'_\beta, s'_x, s'_{\delta_1}, s'_{\delta_2})$ where
    $c' \leftarrow H(M', T_1, T_2, T_3, R'_1, R'_2, R'_3, R'_4, R'_5) \in \mathbb{Z}_p$;

Note that it can demonstrable that the several signatures are produced by the same user, as the information (T_1, T_2, T_3) is public in the same signature. In addition to that, the random values $(r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2})$ must be different than in previous times, that is: $({r_\alpha}' \neq r_\alpha, {r_\beta}' \neq r_\beta, {r_x}' \neq r_x, {r_{\delta_1}}' \neq r_{\delta_1}, {r_{\delta_2}}' \neq r_{\delta_2})$ in order not to reveal information.

D.2.2 Linkability between signatures - Procedure $VerifyLinkable_G$

We define also a new procedure: $VerifyLinkable_G(\sigma, \sigma')$. This algorithm takes as input two signatures

$$\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$$

and

$$\sigma' = ({T_1}', {T_2}', {T_3}', c', s'_\alpha, s'_\beta, s'_x, s'_{\delta_1}, s'_{\delta_2})$$

and outputs *true* or $false$ depending on if the signatures have been produced by the same signer's pseudonym (T_1, T_2, T_3):

$$(T_1 \stackrel{?}{=} T_1', T_2 \stackrel{?}{=} T_2', T_3 \stackrel{?}{=} T_3')$$

D.3 Requirements of the Fare Collection Systems

D.3.1 Common Security Requirements

Transport services give a receipt or a ticket to users in order to be further verified; then, this receipt is a proof that the protocol was followed correctly. In these electronic systems, the following security requirements have to be guaranteed:

- Authenticity: a ticket must be generated by its authorized issuer.
- Non-repudiation: the issuer can not deny the emission of one of its tickets.
- Integrity: the ticket, once generated, can not be further modified.

In addition to these basic requirements, the following ones must be also guaranteed:

- Validity time: Any ticket has a validity time parameter to check whether it is in force or not. Each spent ticket is stored in a database until its validity time has expired.
- Non-overspending: a ticket can only be used once. Before allowing the use of any ticket, its validity period is checked. If this verification is correct, the system checks that the ticket is not in the database of spent tickets by using its serial number. This verification ensures that the ticket is not used more than once.
- Revocable anonymity: the system must guarantee the user's anonymity in order to receive acceptance of the user community, but the system and the public authorities prefer non-anonymity for security and control reasons. Thus, an intermediate solution is revocable anonymity for users. If a user misbehaves, her anonymity is revoked.
- Non-traceability: the provider can only trace an entrance of a user with its corresponding exit, but can never trace different journeys of a same user, what could enable profiles generation.

D.3.2 Requirements for Time-Based Systems

Time-Based fares are most appropriate in environments where the most relevant parameter of the service given is the time. Some good examples of that applied to the transport systems are: taxi services and parking places services. Thus, Time-Based pricing approaches will require time accounts rather than pay per boarding structures. So, in this case, the system has to:

- Create a proper timestamp when a new ticket is issued
- This timestamp creates a time-window where the user has the right to use the service

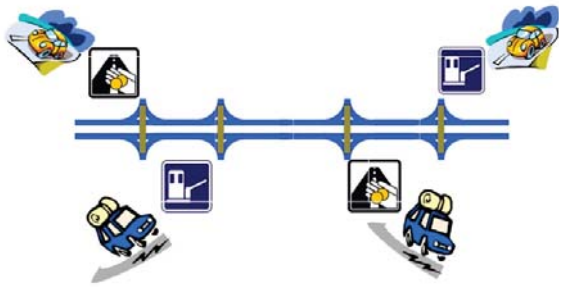


Figure 13: Separated directions in a compliant distance-based AFC.

- The time-window has an initial-date and a expiry-date which determines the maximum period of the service
- The fare to be paid is proportional to the period of time that the costumer has used the service. The longer is the period the higher is the fare
- The timestamp must be checked at the system exit in order to compute the service fare

D.4 Requirements for Distance-Based Systems

Distance-based systems, as described in §D.1.2 calculate the fare to be paid as a function of the entrance and the exit point. The system has to:

- Include the identifier of the entrance station in the entrance ticket.
- Define a validity period for each ticket.
- The fare to be paid is proportional to the distance covered by the user. The larger is the distance, the higher is the fare.

In distance-based systems, services where the fare to be paid depends on the distance, beside the entrance point, the entrance ticket must include a new item. This item is the direction of the movement of the user. If the service checks the direction included in the entrance ticket when the user exits at the destination station the confabulated attacks can be prevented if the system fulfills the following requirements:

- At the entrance station the users must obtain the entrance ticket at different points according to their direction, that is, entrances are separated by direction.
- Users are not able to change the direction of the movement without exiting the service. When a user arrives at the destination station, he must check out according to their direction. That is, system exits are separated by direction.

If the service is non-compliant with the previous requirements, then a more complex protocol will be required to solve the problems appeared due to the possibility of confabulated attacks. The solution for compliant services is described in §D.6.1 while the solution for non-compliant services is included in §D.6.4.

D.5 Time-Based Fare Collection Protocol

In this section, we describe our Time-Based Fare Collection system that provides anonymity to the users by the use of group signatures [48] for mass-transport services. We describe the parties involved in the system, the security requirements to be guaranteed, the information which is contained into the entrance and exit tickets, and finally the phases in which the system consists of.

D.5.1 System Participants

The following actors are involved in the proposed system:

- User $\mathcal{U}$: accesses to the transport system and pays for the received service at the exit. $\mathcal{U}$ performs these actions by means of her mobile device.
- Service provider ($\mathcal{P}_{\mathcal{S}}$ source station, $\mathcal{P}_{\mathcal{D}}$ destination station): checkpoint that controls the tickets used by $\mathcal{U}$. The fare to be paid by $\mathcal{U}$ is computed by $\mathcal{P}_{\mathcal{D}}$ according to the parameters established (time-based or distance-based fares)
- Payment TTP $\mathcal{M}_{\mathcal{C}}$: manages all the user's payments when they exit from the system.
- Group TTP $\mathcal{M}_{\mathcal{G}}$: manages the group keys and the revocation list. It can revoke the user's anonymity in case of misbehaving.

D.5.2 Ticket information

In this section, we describe the information that is included in the entrance ticket at Table 8 and the information in the exit ticket at Table 9. To give a description of the protocols we use the notation described at Table 10.

D.5.3 Protocol Specification

Phases In the protocol, there are the following phases:

- Setup: $\mathcal{M}_{\mathcal{G}}$ generates all the group keys, revocation lists, etc.
- User Registration: $\mathcal{U}$ registers at $\mathcal{M}_{\mathcal{G}}$, she receives a group key pair. $\mathcal{U}$ also registers at $\mathcal{M}_{\mathcal{C}}$ through a pseudonym that will be used only for payments. In the AFC system, $\mathcal{M}_{\mathcal{C}}$ is an entity that establishes user's and service provider's accounts. This entity processes the payment related messages and guarantees the payment for authorized transactions according to the protocol specifications.
- System entrance: the user joins in the source station and generates a group signature that certifies that she is a valid system group member, while her identity is not disclosed. When this signature is sent to the service provider $\mathcal{P}_{\mathcal{S}}$, she receives an entrance ticket from $\mathcal{P}_{\mathcal{S}}$ that will have to be showed in the destination station.

- System exit: the user performs a weak authentication to the destination checkpoint $\mathcal{P}_{\mathcal{D}}$ and shows the entrance ticket. $\mathcal{P}_{\mathcal{D}}$ then calculates the fare to be paid. The user has to accept the fare and sends this information securely to $\mathcal{M}_C$ with his payment pseudonym authentication (only $\mathcal{M}_C$ has knowledge of this pseudonym, $\mathcal{P}_{\mathcal{D}}$ can not disclose that information). Then, $\mathcal{M}_C$ charges the fare to $\mathcal{U}$'s account. If all the process is correct, the user receives an exit ticket, which is an evidence that proves that the user has followed the protocol correctly.

Setup This phase is executed once at first. $\mathcal{M}_{\mathcal{G}}$ executes $KeyGen_G(n)$ which generates a group of preset size n, and outputs $(gpk, gsk[\,], grt[\,], \alpha, p, q)$, where gpk is the common group public key, $gsk[i]$ is the private key for each user $\mathcal{U}_i$, $grt[\,]$ is the revocation list, and (α, p, q) are public parameters, where α is the public exponentiation base, and (p, q) prime numbers where $p = 2q+1$, and they are cardinals of their corresponding groups $\mathbb{Z}_p$ and $\mathbb{Z}_q$. Moreover, each service provider generates its key pair and shows its public key. The private group keys $gsk[i]$ are issued when users are registered at the group.

User Registration $\mathcal{U}$ registers at the group TTP $\mathcal{M}_{\mathcal{G}}$ and receives the group key pair $(gpk, gsk[i])$. At this point, the users agree that their identity will be disclosed if they are not honest, or if a judge requires to revoke their anonymity.

Next, $\mathcal{U}$ also registers anonymously to the payment TTP $\mathcal{M}_C$ with the authorization of $\mathcal{M}_{\mathcal{G}}$; the user owns a pseudonym $y_{\mathcal{U}}$ which is an exponentiation of a random value $x_{\mathcal{U}} \overset{R}{\leftarrow} \mathbb{Z}_q$, where $y_{\mathcal{U}} = \alpha^{x_{\mathcal{U}}} \pmod{p}$; only this information $y_{\mathcal{U}}$ will be showed to $\mathcal{M}_C$ and authenticated through Schnorr's Zero-Knowledge Proof [36] proving knowledge of $x_{\mathcal{U}}$ without disclosing that secret. Thus, privacy is preserved for users, but this anonymity could be revoked by $\mathcal{M}_{\mathcal{G}}$ if necessary. The user registration protocol is defined as follows:

generatePseudonym: The user $\mathcal{U}$ computes:

1. generates her payment pseudonym as a random value $x_{\mathcal{U}} \overset{R}{\leftarrow} \mathbb{Z}_q$;
2. computes $y_{\mathcal{U}} = \alpha^{x_{\mathcal{U}}} \pmod{p}$;
3. sends her identity $\mathcal{U}_i$, her certificate $Cert\mathcal{U}_i$ and a signed message containing the pseudonym
   $Sign_{\mathcal{U}}(y_{\mathcal{U}}, \textit{'Hello'})$ to the Group TTP $\mathcal{M}_{\mathcal{G}}$;

keyIssue: $\mathcal{M}_{\mathcal{G}}$ sends the group key pair $(gpk, gsk[i])$ together with the public parameters (α, p, q) and the signature $Sign_{\mathcal{M}_{\mathcal{G}}}(y_{\mathcal{U}})$ to $\mathcal{U}$;

startingZKP: $\mathcal{U}$ performs:

1. generates a random value $r_0 \overset{R}{\leftarrow} \mathbb{Z}_q$;
2. computes $s_0 = \alpha^{r_0} \pmod{p}$;
3. sends $(y_{\mathcal{U}}, s_0, Sign_{\mathcal{M}_{\mathcal{G}}}(y_{\mathcal{U}}))$ to the Payment TTP $\mathcal{M}_C$;

challengeGeneration: $\mathcal{M}_C$ generates a challenge value $c_0 \overset{R}{\leftarrow} \mathbb{Z}_q$ and sends it to $\mathcal{U}$;

proofGeneration: $\mathcal{U}$ computes the Schnorr's ZKP proof $\omega_0 = r_0 + c_0 \cdot x_{\mathcal{U}} \pmod{q}$ and sends it to $\mathcal{M}_C$;

verifyPseudonym: $\mathcal{M}_C$ verifies that $\alpha^{\omega_0} \stackrel{?}{=} s_0 \cdot (y_{\mathcal{U}})^{c_0}$.

System entrance When $\mathcal{U}$ has correctly entered the system, an entrance ticket $\mathsf{t_{in}}$ is then received. $\mathsf{t_{in}}$ will be later used in order to authorize the user to pay the calculated fee. The system entrance protocol is defined as follows:

getService: The user $\mathcal{U}$ performs:

1. generates a random value $r_1 \stackrel{R}{\leftarrow} \mathbb{Z}_q$;
2. computes $s_1 = \alpha^{r_1} \pmod{p}$;
3. computes $\delta_{\mathcal{U}} = PK_{\mathcal{M}_C}(y_{\mathcal{U}})$ [13];
4. generates a random value $k \stackrel{R}{\leftarrow} \mathbb{Z}_q$;
5. computes the *hash()* function of k: $h_k = hash(k)$;
6. composes $\sigma = (s_1, \delta_{\mathcal{U}}, h_k)$, and signs it with $gsk[i]$, her private group key: $\sigma^* = (\sigma, \bar{\sigma} = Sign_G(gpk, gsk[i], \sigma))$;
7. sends σ^* to $\mathcal{P}_{\mathcal{S}}$;

generateTicket: The source service provider $\mathcal{P}_{\mathcal{S}}$ computes:

1. verifies the signature of σ^*; this entails to check if the signer is a valid group member:
   $Verify_G(gpk, \sigma, \bar{\sigma})$;
2. generates a timestamp τ_1;
3. composes the entrance ticket $\mathsf{t_{in}} = (\mathsf{Sn}, \mathsf{Ps}, \tau_1, \sigma^*)$ and signs it $\mathsf{t_{in}}^* = (\mathsf{t_{in}}, Sign_{\mathcal{P}_{\mathcal{S}}}(\mathsf{t_{in}}))$;
4. sends $\mathsf{t_{in}}^*$ to $\mathcal{U}$;

verifyEntrance: $\mathcal{U}$ verifies the signature of $\mathsf{t_{in}}^*$ and its content;

System exit When the user exits the system, then sends the ticket entrance $\mathsf{t_{in}}$ to the destination service provider $\mathcal{P}_{\mathcal{D}}$, and the fare to be paid is calculated. If $\mathcal{U}$ behaves correctly, an exit ticket $\mathsf{t_{out}}$ is received, and can be later showed as a receipt, entailing that the protocol had been followed correctly. The system exit protocol is defined as follows:

showTicket: $\mathcal{U}$ encrypts k and sends $(\mathsf{t_{in}}^*, PK_{\mathcal{P}_{\mathcal{D}}}(k))$ to $\mathcal{P}_{\mathcal{D}}$;
verifyTicket: The destination service provider $\mathcal{P}_{\mathcal{D}}$ performs:

1. verifies the signature of $\mathsf{t_{in}}^*$ which is computed by $\mathcal{P}_{\mathcal{S}}$;
2. verifies that $\sigma.h_k \stackrel{?}{=} hash(k)$, what proves that $\mathcal{U}$ is the right holder of the ticket $\mathsf{t_{in}}$;
3. verifies that $\mathsf{t_{in}}.\mathsf{Sn}$ had not been previously used;
4. generates a timestamp τ_2 (obviously $\tau_1 \leq \tau_2$);
5. calculates the fare to be paid depending on the elapsed time between corresponding timestamps (τ_1, τ_2): $a = f_i(\mathsf{t_{in}}.\mathsf{Ps}, \mathsf{Pd}, \mathsf{t_{in}}.\tau_1, \tau_2)$;
   Therefore, in this case, $f_i()$ is a function specially designed for computing the fare between two stations on a time-based fare system.

[13] The cryptosystem is probabilistic

6. generates a challenge $c_1 \xleftarrow{R} \mathbb{Z}_q$;
7. composes $\beta = (\mathsf{t_{in}}^*, k, a, c_1, \tau_2, \mathsf{Pd})$, and signs it $\beta^* = (\beta, Sign_{\mathcal{P}_\mathcal{D}}(\beta))$;
8. sends β^* to $\mathcal{U}$ (in case of dispute β can be used by $\mathcal{U}$ as an evidence to prove that she has exit at τ_2- see claim 2);
9. composes $\gamma_{\mathcal{P}_\mathcal{D}} = (\beta.a, \mathsf{t_{in}}.\mathsf{Sn}, \mathsf{t_{in}}.\sigma, c_1)$;

setPayment: $\mathcal{U}$ computes:

1. verifies the signature of β^* which is computed by $\mathcal{P}_\mathcal{D}$;
2. computes $\omega_1 = r_1 + c_1 \cdot x_\mathcal{U} \pmod q$;
3. composes and encrypts $\gamma_\mathcal{U} = PK_{\mathcal{M}_C}(\omega_1, \mathsf{t_{in}}.\mathsf{Sn}, \beta.a)$;
4. sends $\gamma_\mathcal{U}$ to $\mathcal{P}_\mathcal{D}$;

sendingPaymentInfo: $\mathcal{P}_\mathcal{D}$ resends $\gamma_\mathcal{U}$ and $\gamma_{\mathcal{P}_\mathcal{D}}$ to the payment TTP $\mathcal{M}_C$;
verifyPayment: $\mathcal{M}_C$ performs:

1. decrypts $\gamma_\mathcal{U}$ in order to obtain the Schnorr's proof ω_1;
2. decrypts $\mathsf{t_{in}}.\sigma.\delta_\mathcal{U}$ in order to obtain the pseudonym $y_\mathcal{U}$ and charge the fee to the corresponding user's account;
3. verifies the identity of $\mathcal{U}$ through Schnorr's ZKP: $\alpha^{\omega_1} \stackrel{?}{=} s_1 \cdot (y_\mathcal{U})^{c_1}$;
4. if it is correct, the fare a is charged to user's account that possesses $y_\mathcal{U}$ and the protocol continues. Otherwise composes a payment rejection $ko =$ ('authentication error', $\gamma_\mathcal{U}$), signs it $ko^* = (ko, Sign_{\mathcal{M}_C}(ko))$, sends it to $\mathcal{P}_\mathcal{D}$ and stops the protocol;
5. composes $ok = (\mathsf{t_{in}}.\mathsf{Sn}, \beta.a,$ 'ok') and signs it $ok^* = (ok, Sign_{\mathcal{M}_C}(ok))$;
6. sends ok^* to $\mathcal{P}_\mathcal{D}$;

setExit: $\mathcal{P}_\mathcal{D}$ computes:

1. composes $\mathsf{t_{out}} = (\mathsf{t_{in}}.\mathsf{Sn}, \mathsf{Pd}, \beta.a,$ 'leave taking at $\beta.\tau_2$') and signs it $\mathsf{t_{out}}^* = (\mathsf{t_{out}}, Sign_{\mathcal{P}_\mathcal{D}}(\mathsf{t_{out}}))$;
2. sends $\mathsf{t_{out}}^*$ to $\mathcal{U}$ and allows her to exit the system successfully;

checkTicket: $\mathcal{U}$ verifies the signature of $\mathsf{t_{out}}^*$ and its content.

D.5.4 User's Claims

During the *System exit* protocol, $\mathcal{P}_\mathcal{D}$ could not follow the protocol due to different reasons (i.e. $\mathcal{P}_\mathcal{D}$ may fail, make mistakes, crash or commit dishonest actions). Because of that, the honest user would receive an improper service. To solve this problem, our protocol can face two user's claims.

Claim 1: an incorrect β^* is received During the *System exit* protocol, $\mathcal{U}$ can send the validation information $(\mathsf{t_{in}}^*, k)$, but $\mathcal{P}_\mathcal{D}$ could misbehave and sends a wrong β^* (e.g. the message has an inaccurate τ_2) to $\mathcal{U}$ or, simply, $\mathcal{P}_\mathcal{D}$ doesn't send it. Then, this user can claim to receive a valid β^* to Payment TTP $\mathcal{M}_C$ by following these steps:

claim1Request: The user $\mathcal{U}$ resends $(\mathsf{t_{in}}^*, k)$ and the incorrect β^* (if this is the case) to $\mathcal{M}_C$;
claim1Response: The Payment TTP $\mathcal{M}_C$ computes:

1. verifies the signature of $\mathsf{t_{in}}^*$ which is computed by $\mathcal{P}_\mathcal{S}$;
2. verifies that $\sigma.h_k \stackrel{?}{=} hash(k)$, what proves that $\mathcal{U}$ is the right holder of the ticket $\mathsf{t_{in}}$;
3. in case of an incorrect β^*, $\mathcal{M}_C$ verifies that the parameters $\beta.\tau_2$ or $\beta.a$ are not right (e.g. $\beta.\tau_2$ is greater than the current time)
4. generates a new timestamp τ_2. This τ_2 have to represent a slightly reduced time than the current time. $\mathcal{M}_C$ can do that in order to compesate the user due to the time overhead produced by the present transaction in relation to the time when the system exit subprotocol was executed;
5. calculates the fare to be paid depending on the elapsed time between corresponding timestamps (τ_1, τ_2): $a = f_t(\mathsf{Pd}, \mathsf{t_{in}}.\mathsf{Ps}, \mathsf{t_{in}}.\tau_1, \tau_2)$;
6. generates a challenge $c_1 \stackrel{R}{\leftarrow} \mathbb{Z}_q$;
7. composes $\beta = (\mathsf{t_{in}}^*, a, c_1, \tau_2, \mathsf{Pd})$, and signs it $\beta^* = (\beta, Sign_{\mathcal{M}_C}(\beta))$;
8. sends β^* to $\mathcal{U}$;

resume: The *System exit* protocol continues normally.

Claim 2: an incorrect $\mathsf{t_{out}}^*$ is received During the *System exit* protocol, $\mathcal{U}$ can send the validation information $(\mathsf{t_{in}}^*, k, \gamma_\mathcal{U})$, but $\mathcal{P}_\mathcal{D}$ could misbehave and sends an incorrect $\mathsf{t_{out}}^*$ to $\mathcal{U}$ or simply denies to send it. Then, the user can contact with the Payment TTP $\mathcal{M}_C$ and she can claim to receive a valid $\mathsf{t_{out}}^*$ by following these steps:

claim2Request: The user $\mathcal{U}$ resends $(\mathsf{t_{in}}^*, k, \beta^*, \gamma_\mathcal{U})$ to $\mathcal{M}_C$;
claim2Response: The Payment TTP $\mathcal{M}_C$ computes:

1. verifies the signature of $\mathsf{t_{in}}^*$ which was computed by $\mathcal{P}_\mathcal{S}$;
2. verifies the identity of $\mathcal{U}$ through Schnorr's ZKP: $\alpha^{\omega_1} \stackrel{?}{=} s_1 \cdot (y_\mathcal{U})^{c_1}$;
3. verifies that $\sigma.h_k \stackrel{?}{=} hash(k)$, what proves that $\mathcal{U}$ is the right holder of the ticket $\mathsf{t_{in}}$;
4. calculates the fare to be paid depending on the elapsed time between corresponding timestamps (τ_1, τ_2) : $a = f_t(\mathsf{t_{in}}.\mathsf{Ps}, \beta.\mathsf{Pd}, \mathsf{t_{in}}.\tau_1, \beta.\tau_2)$. Then, $\mathcal{M}_C$ verifies that the calculated value a is equal to $\beta.a$. In case of a negative verification, the user is addressed to execute the protocol specified at claim 1;
5. composes $\mathsf{t_{out}} = (\mathsf{t_{in}}.\mathsf{Sn}, \beta.a$, 'leave taking at $\beta.\tau_2$'), and signs it $\mathsf{t_{out}}^* = (\mathsf{t_{out}}, Sign_{\mathcal{M}_C}(\mathsf{t_{out}}))$;
6. sends $\mathsf{t_{out}}^*$ to $\mathcal{U}$;

resume: The *System exit* protocol continues normally.

In both claims, the Payment TTP $\mathcal{M}_C$ has to warn $\mathcal{P}_\mathcal{D}$ of its misbehavior or communication problems with users. $\mathcal{M}_C$ also has to alert $\mathcal{P}_\mathcal{D}$ of possible further actions if this problem persists.

D.5.5 Provider's Claims

During the *System exit* protocol, $\mathcal{U}$ could not follow the protocol due to different reasons (i.e. $\mathcal{U}$ may fail, make mistakes, crash or commit **dishonest actions**). Because

of that, the provider would receive an improper service. To solve this problem, our protocol can face two provider's claims.

Claim 3: An incorrect $(\mathsf{t_{in}}^*, k)$ **is received** During the *System exit* protocol, $\mathcal{P}_\mathcal{D}$ receives the first step of the verification information $(\mathsf{t_{in}}^*, k)$, but this information could be not correct, or could not link. Then, this service provider can claim to disclose user's identity by following these steps:

claim3Request: The destination service provider $\mathcal{P}_\mathcal{D}$ sends $(\mathsf{t_{in}}^*, k)$ to $\mathcal{M}_\mathcal{G}$;
appealingUser: The user $\mathcal{U}$ is required to send also $(\mathsf{t_{in}}^*, k)$ to $\mathcal{M}_\mathcal{G}$, in order to avoid false accusations;
claim3Response: If $\mathcal{U}$ does not send the required items, the Group TTP $\mathcal{M}_\mathcal{G}$ computes:

1. verifies the signature of $(\mathsf{t_{in}}^*, k)$ which is generated by $\mathcal{P}_\mathcal{S}$;
2. verifies the link with the hash value $\mathsf{t_{in}}.\sigma.h_k \stackrel{?}{=} hash(k)$; If the link is not verified, $\mathcal{M}_\mathcal{G}$ aborts the claim;
3. verifies the group signature of $\mathsf{t_{in}}.\sigma^*$ which is generated by $\mathcal{U}$, disclosing then who is the signer inside the group;
4. sends the user identification $\mathcal{U}_i$ to $\mathcal{P}_\mathcal{D}$ and $y_\mathcal{U}$ to $\mathcal{M}_\mathcal{C}$;
5. $\mathcal{U}_i$ is added to the revocated list;

Claim 4: An incorrect $\gamma_\mathcal{U}$ **is received** During the *System exit* protocol, $\mathcal{P}_\mathcal{D}$ and $\mathcal{M}_\mathcal{C}$ receives the last step of the verification information $\gamma_\mathcal{U}$, but this information could be not correct. Then, this service provider can claim to disclose user's identity by following these steps:

claim4Request: The Payment TTP $\mathcal{M}_\mathcal{C}$ composes a payment rejection $ko =$ ('verification information error', $\gamma_\mathcal{U}$), signs it $ko^* = (ko, Sign_{\mathcal{M}_\mathcal{C}}(ko))$ and sends it to $\mathcal{P}_\mathcal{D}$. The Payment TTP $\mathcal{M}_\mathcal{C}$ also sends $(sk_{\mathcal{P}_\mathcal{D}}(\gamma_\mathcal{U}), \gamma_{\mathcal{P}_\mathcal{D}})$ to $\mathcal{M}_\mathcal{G}$ and stops the protocol.
providerInfo: The destination service provider $\mathcal{P}_\mathcal{D}$ sends $(\mathsf{t_{in}}^*, k)$ to $\mathcal{M}_\mathcal{G}$;
appealingUser: The user $\mathcal{U}$ is required to send also $(\mathsf{t_{in}}^*, k, \gamma_\mathcal{U})$ to $\mathcal{M}_\mathcal{G}$, in order to avoid false accusations;
claim4Response: If $\mathcal{U}$ does not send the required items, the Group TTP $\mathcal{M}_\mathcal{G}$ computes:

1. verifies if the decrypted information of $\gamma_\mathcal{U}$, $\gamma_{\mathcal{P}_\mathcal{D}}$ and $(\mathsf{t_{in}}^*, k)$ link;
2. verifies the group signature of $\mathsf{t_{in}}.\sigma^*$ which is generated by $\mathcal{U}$, disclosing then who is the signer inside the group;
3. sends the user identification $\mathcal{U}_i$ to $\mathcal{P}_\mathcal{D}$ and $y_\mathcal{U}$ to $\mathcal{M}_\mathcal{C}$;
4. $\mathcal{U}_i$ is added to the revocated list;

D.6 Distance-Based Fare Collection Protocol

In this section, we describe our Distance-Based Fare Collection system that provides anonymity to the users by the use of group signatures [48] for mass-transport services.

We have adapted the Time-Based protocol in §D.5 into a Distance-Based Fare Collection Protocol with little changes. Then, the complexity of the adaptation depends mainly on the degree of the requirements' compliance described in §D.4.

Firstly, we show the changes to be made to the requirement compliant distance-based systems in §D.6.1. Later, in §D.6.2, we describe the services which are non-compliant, introducing then a fraud attack: the colluding attack, in §D.6.3. Finally, in §D.6.4, we describe the changes to be made to the non-compliant distance-based systems.

D.6.1 Requirement Compliant Distance-Based Systems

Little changes have to be made to the protocol presented in §D.5 in order to adapt it to a distance-based automated fare collection system that fulfils the requirements presented in §D.4. We only have to add one item to the information gathered in the entrance ticket. We add the direction of the journey (ξ) and the validity time (τ_v), so now an entrance ticket is: $\mathsf{t_{in}} = (\mathsf{Sn}, \mathsf{Ps}, \tau_1, \tau_v, \sigma^*, \xi)$.

generateTicket: The source service provider $\mathcal{P}_\mathcal{S}$ computes:

1. verifies the signature of σ^*; this entails to check if the signer is a valid group member:
   $Verify_G(gpk, \sigma, \bar{\sigma})$;
2. generates a timestamp τ_1;
3. composes the entrance ticket:
   $\mathsf{t_{in}} = (\mathsf{Sn}, \mathsf{Ps}, \tau_1, \tau_v, \sigma^*, \xi)$ and signs it:
   $\mathsf{t_{in}}^* = (\mathsf{t_{in}}, Sign_{\mathcal{P}_\mathcal{S}}(\mathsf{t_{in}}))$;
4. sends $\mathsf{t_{in}}^*$ to $\mathcal{U}$;

Concerning the protocol specifications, we have only to change the action **verifyTicket** performed by $\mathcal{P}_\mathcal{D}$ in the system exit subprotocol and the **claim1Response** and **claim2Response** because we have to modify the way of calculating the fare according to a distance-base criteria. Then, **verifyTicket** for a distance-based scheme is as follows:

verifyTicket: The destination service provider $\mathcal{P}_\mathcal{D}$ performs:

1. verifies the signature of $\mathsf{t_{in}}^*$ which is computed by $\mathcal{P}_\mathcal{S}$;
2. verifies that $\sigma.h_k \stackrel{?}{=} hash(k)$, what proves that $\mathcal{U}$ is the right holder of the ticket $\mathsf{t_{in}}$;
3. verifies that $\mathsf{t_{in}}.\mathsf{Sn}$ had not been previously used;
4. verifies that the validity time τ_v has not expired, and the direction ξ is correct;
5. generates a timestamp τ_2 (obviously $\tau_1 \leq \tau_2$);
6. the entrance station ($\mathsf{t_{in}}.\mathsf{Ps}$), the exit station ($\mathsf{Pd}$) and their corresponding timestamps (τ_1, τ_2):
   $a = f_d(\mathsf{t_{in}}.\mathsf{Ps}, \mathsf{Pd}, \mathsf{t_{in}}.\tau_1, \tau_2)$;
   Therefore, in this case, $f_d()$ is a function specially designed for computing the fare between two stations on a distance-based fare system.

7. generates a challenge $c_1 \overset{R}{\leftarrow} \mathbb{Z}_q$;
8. composes $\beta = (\mathsf{t_{in}}^*, k, a, c_1, \tau_2, \mathsf{Pd})$, and signs it $\beta^* = (\beta, Sign_{\mathcal{P}_\mathcal{D}}(\beta))$;
9. sends β^* to $\mathcal{U}$ (in case of dispute β can be used by $\mathcal{U}$ as an evidence to prove that she has exit at τ_2- see claim 2);
10. composes $\gamma_{\mathcal{P}_\mathcal{D}} = (\beta.a, \mathsf{t_{in}}.\mathsf{Sn}, \mathsf{t_{in}}.\sigma, c_1)$;

The **claim1Response** function in a distance-based system has to be as follows:

claim1Response: The Payment TTP $\mathcal{M}_C$ computes:

1. verifies the signature of $\mathsf{t_{in}}^*$ which is computed by $\mathcal{P}_\mathcal{S}$;
2. verifies that $\sigma.h_k \overset{?}{=} hash(k)$, what proves that $\mathcal{U}$ is the right holder of the ticket $\mathsf{t_{in}}$;
3. in case of an incorrect β^*, $\mathcal{M}_C$ verifies that the parameters $\beta.\tau_2$ or $\beta.a$ are not right (e.g. $\beta.\tau_2$ is greater than the current time)
4. verifies that the validity time τ_v has not expired, and the direction ξ is correct;
5. generates a new timestamp τ_2.
6. calculates the fare to be paid depending on the entrance station ($\mathsf{t_{in}}.\mathsf{Ps}$) and the exit station ($\mathsf{Pd}$):
   $a = f_d(\mathsf{Pd}, \mathsf{t_{in}}.\mathsf{Ps}, \mathsf{t_{in}}.\tau_1, \tau_2)$;
7. generates a challenge $c_1 \overset{R}{\leftarrow} \mathbb{Z}_q$;
8. composes $\beta = (\mathsf{t_{in}}^*, a, c_1, \tau_2, \mathsf{Pd})$, and signs it $\beta^* = (\beta, Sign_{\mathcal{M}_C}(\beta))$;
9. sends β^* to $\mathcal{U}$;

Finally, the **claim2Response** has to be as follows:

claim2Response: The Payment TTP $\mathcal{M}_C$ computes:

1. verifies the signature of $\mathsf{t_{in}}^*$ which was computed by $\mathcal{P}_\mathcal{S}$;
2. verifies the identity of $\mathcal{U}$ through Schnorr's ZKP: $\alpha^{\omega_1} \overset{?}{=} s_1 \cdot (y_\mathcal{U})^{c_1}$;
3. verifies that $\sigma.h_k \overset{?}{=} hash(k)$, what proves that $\mathcal{U}$ is the right holder of the ticket $\mathsf{t_{in}}$;
4. verifies that the validity time τ_v has not expired, and the direction ξ is correct;
5. calculates the fare to be paid depending on the entrance station ($\mathsf{t_{in}}.\mathsf{Ps}$) and the exit station ($\beta.\mathsf{Pd}$): $a = f_d(\mathsf{t_{in}}.\mathsf{Ps}, \beta.\mathsf{Pd}, \mathsf{t_{in}}.\tau_1, \beta.\tau_2)$. Then, $\mathcal{M}_C$ verifies that the calculated value a is equal to $\beta.a$. In case of a negative verification, the user will be addressed to execute the claim 1 subprotocol;
6. composes $\mathsf{t_{out}} = (\mathsf{t_{in}}.\mathsf{Sn}, \beta.a$, 'leave taking at $\beta.\tau_2$'), and signs it $\mathsf{t_{out}}^* = (\mathsf{t_{out}}, Sign_{\mathcal{M}_C}(\mathsf{t_{out}}))$;
7. sends $\mathsf{t_{out}}^*$ to $\mathcal{U}$;

D.6.2 Non-compliant distance-based services

In this section, we treat the services which depend on distance and do not comply with the requirements. Users could access and exit the system in different points, but now

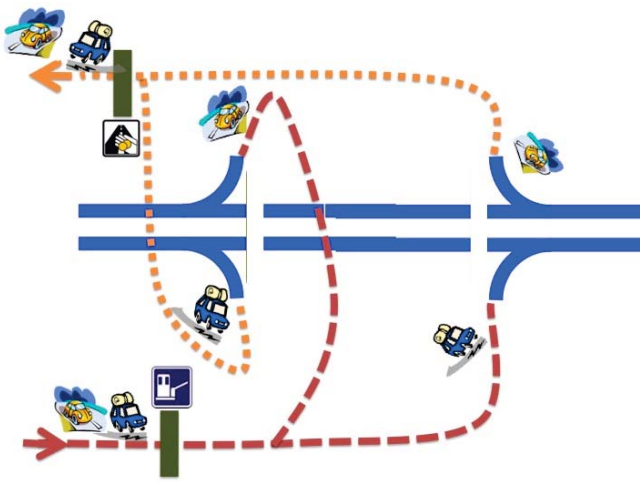


Figure 14: Non-compliant distance-based AFC.

not separating them by their direction. That is, maybe in some parts of the system entrances and exits are indistinguishable in this point of view, as they have common areas. With this situation, a major complexity in the transport system could open security holes and require then deeper security measures. We describe a possible colluding attack in the next section.

D.6.3 Colluding Attacks

Here we address the problem where, in a distance-based fare collection system, it is not sure that the entrances and exits in the stations are distinguishable. If the stations do not differ their exits depending on the direction, then the colluding attacks could be easily performed without detection. Imagine that a determined user $\mathcal{U}_1$ joins the system in a determined station $\mathcal{P}_{\mathcal{S}1}$ and exits the system in $\mathcal{P}_{\mathcal{D}1}$, and another $\mathcal{U}_2$ joins in $\mathcal{P}_{\mathcal{S}2}$ and exits in $\mathcal{P}_{\mathcal{D}2}$. As consequently, they would have to pay their fares depending on the differences between the stations distance, that is: $f_{D_1}(\mathcal{P}_{\mathcal{S}1}, \mathcal{P}_{\mathcal{D}1}, \tau_{11}, \tau_{12})$ and $f_{D_2}(\mathcal{P}_{\mathcal{S}2}, \mathcal{P}_{\mathcal{D}2}, \tau_{21}, \tau_{22})$. In this scenario, the users could collaborate in some advantaging cases in order both to pay less if they exchanged their received entrance tickets; that is: $f_{D_1}{}'(\mathcal{P}_{\mathcal{S}2}, \mathcal{P}_{\mathcal{D}1}, \tau_{21}, \tau_{12})$ and $f_{D_2}{}'(\mathcal{P}_{\mathcal{S}1}, \mathcal{P}_{\mathcal{D}2}, \tau_{11}, \tau_{22})$. For a graphical explanation, see Fig. 14, where we can see that the cars traveling in opposite directions can enter the system through the same station. After obtaining a ticket they are able to choose the direction. Similarly, they exit the system using the same provider. The provider is not able to determine which was the direction of the car. This fact can be used by users to try confabulated attacks.

With the current system, it would be quite easy to perform it without detection, as the entrance and exit tickets are not *hard-linked* between them. In Fig. 15, we have analyzed the fraud ratio, regarding the cases where users could take advantage of this

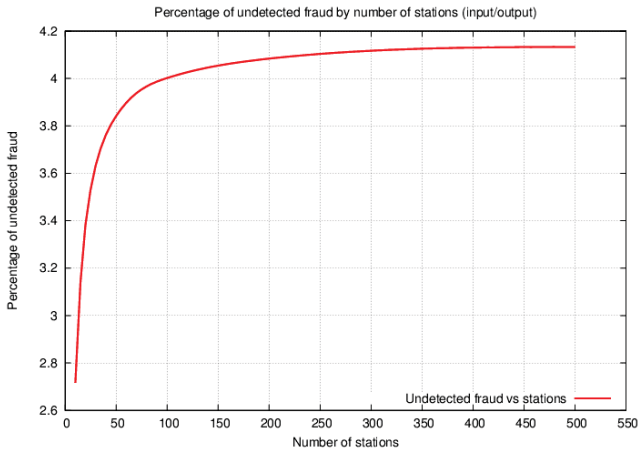


Figure 15: Percentage of undetected fraud by number of stations in a linear scenario (in %)

exchange and would be not detectable with the previous system. The scenario is a lineal sort of stations with only common areas of entrances/exits, and we evaluate this ratio depending on the number of stations.

In the next section, we propose a system that can be used in order to detect this colluding attack.

D.6.4 Distance-based fare collection for non-compliant services

Our intention is to produce a hard link between the entrance ticket and the proofs at the exit. The major change is that the k parameter and its hash function have been replaced for a second signature in the exit which is linkable to the signature in the entrance (see §D.2 for details). Taking into account this modification, the security is incremented, but their computational costs will be higher. An analysis and comparison of the computational times is presented in §D.8. We describe here only the changes to be applied to the current protocol.

The *System entrance* changes in the protocol are:

getService: The user $\mathcal{U}$ performs:

1. generates a random value $r_1 \overset{R}{\leftarrow} \mathbb{Z}_q$;
2. computes $s_1 = \alpha^{r_1} \pmod{p}$;
3. computes $\delta_{\mathcal{U}} = PK_{\mathcal{M}_C}(y_{\mathcal{U}})$ [14];
5. sends σ^* to $\mathcal{P}_{\mathcal{S}}$;

generateTicket: The source service provider $\mathcal{P}_{\mathcal{S}}$ computes:

[14] The cryptosystem is probabilistic;

4. composes $\sigma = (s_1, \delta_{\mathcal{U}})$, and signs it with $gsk[i]$, her private group key: $\sigma^* = (\sigma, \bar{\sigma} = Sign_G(gpk, gsk[i], \sigma))$

1. verifies the signature of σ^*; this entails to check if the signer is a valid group member:
   $Verify_G(gpk, \sigma, \bar{\sigma})$;
2. generates a timestamp τ_1;
3. composes the entrance ticket
   $\mathsf{t_{in}} = (\mathsf{Sn}, \mathsf{Ps}, \tau_1, \tau_v, \sigma^*, \xi)$ and signs it:
   $\mathsf{t_{in}}^* = (\mathsf{t_{in}}, Sign_{\mathcal{P}_S}(\mathsf{t_{in}}))$;
4. sends $\mathsf{t_{in}}^*$ to $\mathcal{U}$;

The modification of the *System exit* protocol is as follows:

previousStep: $\mathcal{P}_\mathcal{D}$ generates a $\Phi \xleftarrow{R} \mathbb{Z}_p$ value and sends it to $\mathcal{U}$;
showTicket: $\mathcal{U}$ computes:

1. signs the received Φ as the same member than in the entrance: $\Phi^* = (\Phi, \bar{\Phi})$ where $\bar{\Phi} = SignLinkable_G(gpk, gsk[i], \Phi)$);
2. sends $(\mathsf{t_{in}}^*, \Phi^*))$ to $\mathcal{P}_\mathcal{D}$;

verifyTicket: The destination service provider $\mathcal{P}_\mathcal{D}$ performs:

1. verifies the signature of $\mathsf{t_{in}}^*$ which is computed by $\mathcal{P}_\mathcal{S}$;
2. verifies the group signature of Φ^*:
   $(Verify_G(gpk, \Phi, \bar{\Phi}))$ and that is the same member than in the entrance:
   $VerifyLinkable_G(\mathsf{t_{in}}.\sigma^*, \Phi^*)$;
3. verifies that $\mathsf{t_{in}}.\mathsf{Sn}$ had not been previously used;
4. verifies that the validity time τ_v has not expired, and the direction ξ is correct;
5. generates a timestamp τ_2 (obviously $\tau_1 \leq \tau_2$);
6. calculates the fare to be paid depending on the entrance station ($\mathsf{t_{in}}.\mathsf{Ps}$), the exit station ($\mathsf{Pd}$) and their corresponding timestamps (τ_1, τ_2): $a = f_d(\mathsf{t_{in}}.\mathsf{Ps}, \mathsf{Pd}, \mathsf{t_{in}}.\tau_1, \tau_2)$;
7. generates a challenge $c_1 \xleftarrow{R} \mathbb{Z}_q$;
8. composes $\beta = (\mathsf{t_{in}}^*, a, c_1, \tau_2, \mathsf{Pd})$, and signs it $\beta^* = (\beta, Sign_{\mathcal{P}_\mathcal{D}}(\beta))$;
9. sends β^* to $\mathcal{U}$ (in case of dispute β can be used by $\mathcal{U}$ as an evidence to prove that she has exit at τ_2- see claim 2);
10. composes $\gamma_{\mathcal{P}_\mathcal{D}} = (\beta.a, \mathsf{t_{in}}.\mathsf{Sn}, \mathsf{t_{in}}.\sigma, c_1)$;

The changes in *Claim 1* are:

claim1Request: The user $\mathcal{U}$ resends $(\mathsf{t_{in}}^*, \Phi^*)$ and the incorrect β^* (if this is the case) to $\mathcal{M}_C$;
claim1Response: The Payment TTP $\mathcal{M}_C$ computes:

1. verifies the signature of $\mathsf{t_{in}}^*$ which is computed by $\mathcal{P}_\mathcal{S}$;
2. verifies the group signature of Φ^*:
   $(Verify_G(gpk, \Phi, \bar{\Phi}))$ and that is the same member than in the entrance:
   $VerifyLinkable_G(\mathsf{t_{in}}.\sigma^*, \Phi^*)$;
3. verifies that the validity time τ_v has not expired, and the direction ξ is correct;

4. in case of an incorrect β^*, $\mathcal{M}_C$ verifies that the parameters $\beta.\tau_2$ or $\beta.a$ are not right (e.g. $\beta.\tau_2$ is greater than the current time)
5. generates a new timestamp τ_2. This τ_2 have to represent a slightly reduced time than the current time. $\mathcal{M}_C$ can do that in order to compesate the user due to the time overhead produced by the present transaction in relation to the time when the system exit subprotocol was executed;
6. calculates the fare to be paid depending on the entrance station ($\mathsf{t_{in}}.\mathsf{Ps}$), the exit station ($\mathsf{Pd}$) and their corresponding timestamps (τ_1, τ_2): $a = f_d(\mathsf{t_{in}}.\mathsf{Ps}, \mathsf{Pd}, \mathsf{t_{in}}.\tau_1, \tau_2)$;
7. generates a challenge $c_1 \overset{R}{\leftarrow} \mathbb{Z}_q$;
8. composes $\beta = (\mathsf{t_{in}}^*, a, c_1, \tau_2, \mathsf{Pd})$, and signs it $\beta^* = (\beta, Sign_{\mathcal{M}_C}(\beta))$;
9. sends β^* to $\mathcal{U}$;

The changes in *Claim 2* are defined as follows:

claim2Request: The user $\mathcal{U}$ resends $(\mathsf{t_{in}}^*, \Phi^*, \beta^*, \gamma_{\mathcal{U}})$ to $\mathcal{M}_C$;
claim2Response: The Payment TTP $\mathcal{M}_C$ computes:

1. verifies the signature of $\mathsf{t_{in}}^*$ which was computed by $\mathcal{P}_S$;
2. verifies that the validity time τ_v has not expired, and the direction ξ is correct;
3. verifies the identity of $\mathcal{U}$ through Schnorr's ZKP: $\alpha^{\omega_1} \overset{?}{=} s_1 \cdot (y_{\mathcal{U}})^{c_1}$;
4. verifies the group signature of Φ^*:
   $(Verify_G(gpk, \Phi, \bar{\Phi}))$ and that is the same member than in the entrance:
   $VerifyLinkable_G(\mathsf{t_{in}}.\sigma^*, \Phi^*)$;
5. calculates the fare to be paid depending on the entrance station ($\mathsf{t_{in}}.\mathsf{Ps}$), the exit station ($\mathsf{Pd}$) and their corresponding timestamps (τ_1, τ_2): $a = f_d(\mathsf{t_{in}}.\mathsf{Ps}, \mathsf{Pd}, \mathsf{t_{in}}.\tau_1, \tau_2)$. Then, $\mathcal{M}_C$ verifies that the calculated value a is equal to $\beta.a$;
6. composes $\mathsf{t_{out}} = (\mathsf{t_{in}}.\mathsf{Sn}, \beta.a$, 'leave taking at $\beta.\tau_2$'), and signs it $\mathsf{t_{out}}^* = (\mathsf{t_{out}}, Sign_{\mathcal{M}_C}(\mathsf{t_{out}}))$;
7. sends $\mathsf{t_{out}}^*$ to $\mathcal{U}$;

Equally, the changes in *Claim 3* are defined as follows:

claim3Request: The destination service provider $\mathcal{P}_D$ sends $(\mathsf{t_{in}}^*, \Phi^*)$ to $\mathcal{M}_G$;
appealingUser: The user $\mathcal{U}$ is required to send also $(\mathsf{t_{in}}^*, \Phi^*)$ to $\mathcal{M}_G$, in order to avoid false accusations;
claim3Response: If $\mathcal{U}$ does not send the required items, the Group TTP $\mathcal{M}_G$ computes:

1. verifies the signature of $(\mathsf{t_{in}}^*, \Phi^*)$ which is generated by $\mathcal{P}_S$;
2. verifies the group signature of Φ^*:
   $(Verify_G(gpk, \Phi, \bar{\Phi}))$ and that is the same member than in the entrance:
   $VerifyLinkable_G(\mathsf{t_{in}}.\sigma^*, \Phi^*)$;
3. verifies the group signature of $\mathsf{t_{in}}.\sigma^*$ which is generated by $\mathcal{U}$, disclosing then who is the signer inside the group with $Open_G(gpk, gmsk, \mathsf{t_{in}}.\sigma^*)$;
4. sends the user identification $\mathcal{U}_i$ to $\mathcal{P}_D$ and $y_{\mathcal{U}}$ to $\mathcal{M}_C$;

5. $\mathcal{U}_i$ is added to the revocated list;

Finally, the changes in *Claim 4* are defined as follows:

providerInfo: The destination service provider $\mathcal{P}_\mathcal{D}$ sends $(\mathsf{t_{in}}^*, \Phi^*)$ to $\mathcal{M}_\mathcal{G}$;
appealingUser: The user $\mathcal{U}$ is required to send also $(\mathsf{t_{in}}^*, \Phi^*, \gamma_\mathcal{U})$ to $\mathcal{M}_\mathcal{G}$, in order to avoid false accusations;
claim4Response: If $\mathcal{U}$ does not send the required items, the Group TTP $\mathcal{M}_\mathcal{G}$ computes:

1. verifies if the decrypted information of $\gamma_\mathcal{U}$, $\gamma_{\mathcal{P}_\mathcal{D}}$ and $(\mathsf{t_{in}}^*, \Phi^*)$ link;
2. verifies the group signature of $\mathsf{t_{in}}.\sigma^*$ which is generated by $\mathcal{U}$, disclosing then who is the signer inside the group with $Open_G(gpk, gmsk, \mathsf{t_{in}}.\sigma^*)$;
3. sends the user identification $\mathcal{U}_i$ to $\mathcal{P}_\mathcal{D}$ and $y_\mathcal{U}$ to $\mathcal{M}_\mathcal{C}$;
4. $\mathcal{U}_i$ is added to the revocated list;

D.7 Security analysis

Proposition 1 *The proposed system preserves authenticity, non-repudiation and integrity for the entrance and exit tickets.*

Claim 1. *The creation of fraudulent tickets is computationally unfeasible nowadays.*

Proof. On the one hand, the tickets are signed: $\mathsf{t_{in}}^* = (\mathsf{t_{in}}, Sign_{\mathcal{P}_\mathcal{S}}(\mathsf{t_{in}}))$ and $\mathsf{t_{out}}^* = (\mathsf{t_{out}}, Sign_{\mathcal{P}_\mathcal{D}}(\mathsf{t_{out}}))$, and also the sent information before the payment $\beta^* = (\beta, Sign_{\mathcal{P}_\mathcal{D}}(\beta))$. If an unauthorized entity can create a valid ticket (entrance or exit) without knowledge of the private keys of $\mathcal{P}_\mathcal{S}$ nor $\mathcal{P}_\mathcal{D}$, it could generate digital signatures while impersonating these providers. Supposing that we use a secure digital signature scheme, this operation is considered unfeasible. On the other hand, the user sends the verification information signed with her group private key $\sigma^* = (\sigma, Sign_\mathcal{G}(\sigma))$. For the same reason, this signature guarantees that the message is authentic and has been issued by a valid user (and not revoked) inside the group.

Claim 2. *The issuer of a ticket can not deny the emission of this ticket.*

Proof. The tickets are signed by its authorized issuer (service providers) and, by considering that the used signature scheme is secure, this operation could be only performed by these issuers. Thus, the issuer's identity is linked to the ticket and, for the properties of the electronic signature scheme, that issuer can not deny its authorship. The same occurs with the group signature scheme, where if the identity is disclosed, the message authorship can be verified.

Claim 3. *The content of the tickets can not be modified.*

Proof. If we suppose that the signature scheme is secure and the *hash* summary function is collision-resistant and its inverse function is computationally unfeasible nowadays, then, if the ticket content was modified, the verification of the signature would be incorrect. In order to pass the verification, the signature would be needed to be regenerated from the new ticket content. This operation is computationally unfeasible nowadays with the most current machines. The same occurs with the group signature scheme.

Result of Proposition 6 *According to the definitions given at §D.3.1 and the Claims 1, 2 and 3, we can assure that the protocol achieves the security requirements of authenticity, non-repudiation and integrity.*

Proposition 2 *The system described in this paper achieves revocable anonymity for users, and all the movements performed by a same user are untraceable each other if the service providers attempt to trace them.*

Claim 4. *A ticket is anonymous.*

Proof. The information related to the user's identity is encrypted with the payment TTP's public key. The service providers ($\mathcal{P}_S$ and $\mathcal{P}_D$) can not access to this information because they need the private key of the TTP. In the system, users compute a group signature ($t_{in}.\sigma^* = (\sigma, Sign_G(\sigma))$) which certify that the signer is a valid group member. If we analyse the properties of the group signatures scheme, the providers can not disclose the identity of the signature generator. In case of controversial situation, the identity of the user who signed the content could be disclosed through the cooperation of both payment TTP $\mathcal{M}_C$ and the group TTP $\mathcal{M}_G$. If the user appears in the revocation list, her identity is revealed, enabling then further actions.

Claim 5. *The user is anonymous by the service providers during the payment phase.*

Proof. All the information related to the payment is encrypted and only the payment TTP can access to it. The service providers are external to the payment, and only receive the payment confirmation from the payment TTP $\mathcal{M}_C$. Then, $\mathcal{M}_C$ has knowledge about y_U from the pair (x_U, y_U) where $y_U = \alpha^{x_U} \pmod{p}$, which identifies her as a valid user; then, the user authenticates by proving knowledge of x_U through Schnorr's ZKP [36].

Claim 6. *Multiple group signatures performed by the same user must be untraceable one each other by the service providers or other entities external to the system.*

Proof. The group signature proposal [48] by Boneh, Boyen and Shacham uses a probabilistic signature scheme, that is, it is not possible to predict a ciphertext given a certain plaintext. This allows untraceability between different group signatures performed by the same user.

Result of Proposition 7 *According to the definitions given at §D.3.1 and the Claims 4, 5 and 6, we can assure that the protocol achieves the security requirements of revocable anonymity and untraceability.*

Proposition 3 *The protocol avoids ticket overspending and also guarantees the control of the validity times.*

Claim 7. *The protocol avoids ticket overspending.*

Proof. If a user tries to overspend an entrance ticket, the serial number will be marked as already used. If this user misbehaviour can be proved, the group TTP $\mathcal{M}_G$ could include this user to the revocation list.

Claim 8. *The ticket can not be further valid if its validity time τ_v has expired.*

Proof. The destination station $\mathcal{P}_D$ receives the ticket from the user in order to be verified. In this verification, the current time is compared to the validity time τ_v of the entrance ticket $t_{in}{}^*$ which is signed by $\mathcal{P}_S$.

Result of Proposition 8 *According to the definitions given at §D.3.1 and the Claims 7 and 8, we can assure that the protocol achieves the security requirements of non-overspending and the control of the validity time of the ticket.*

Proposition 4 *The proposed protocols avoid attacks made by confabulated users.*

CLAIM 9. *The time-based fare Collection system described in §D.5 cannot be attacked by confabulated users.*

PROOF. The confabulated attack described in §D.6 based in the exchange of entrance tickets is not applicable to time-based Systems since the users do not obtain any benefit of the exchange. The fare is calculated using the entrance timestamp so if the users exchange their tickets the fares will be the same and one of the users would pay more than with his real ticket. For this reason users are discouraged to exchange tickets.

CLAIM 10. *The distance-based fare collection system described in §D.6.1 cannot be attacked by confabulated users.*

PROOF. In distance-based fare collection systems, an attack based in the exchange of the ticket would be profitable only in some cases. In all of these cases the direction of the users must be different. If the distance-based fare collection system fulfils the requirements and the directions are physically separated then the users traveling in opposite directions would not be able to use an exchanged ticket to exit the system since the direction included in the ticket would be different from the real one of the user.

CLAIM 11. *The protocol presented in §D.6 can be used in all kind of distance-based fare collection system and avoids confabulated attacks.*

PROOF. Distance-based fare collection systems that do not fulfill the requirements require an improved protocol to avoid confabulated attacks. §D.6 includes the improved protocol including a new group signature. With the use of linkable signatures, even anonymously, the provider can assure that the user who exits the system is the one who obtained the entrance ticket. For this reason users cannot attack the system exchanging tickets.

Result of Proposition 9 *According to the attack described in §D.6 and the protocols described in §D.5 and §D.6, we can assure that the protocol cannot be attacked by confabulated users, neither for time-based fare collection systems nor for distance-based fare collection systems.*

D.8 Implementation

In order to evaluate the protocol performance we have implemented the protocol described above. Because the protocol is intended to be used with mobile devices (handsets or in-car devices) it should be implemented in a mobile platform. There are several mobile platforms available but Android seems is the growing one [49, 50] in all type of devices, from low class to high technology terminals. Furthermore, Android uses the well know Java language for the application development, it is supported and actively developed by Google and also by a big users community who can provide a valuable feedback and help.

The first challenge was the implementation of the Short Group Signature by Boneh, Boyen and Shacham. After a deep search on the network, we cannot find any implementation of this scheme written in Java. So we decided to develop our own implementation from scratch. As the scheme is based on bilinear maps and pairings we chose to use the jPBC (Java Based Pairing Cryptography) [51] library. This library is a full Java port of PBC (Pairing Based Cryptography) [52] C library in which Shacham, one of the authors of the group signature, contributed to the development. The jPBC library provides us the ability to compute complex pairing operations over elliptic curves required by the group signature scheme.

As the group signature implementation uses pairing based cryptography and in order to unify the development, we have decided that the randoms, exponentiations and arithmetic operations use this kind of cryptography too. On the other hand, the common signatures and encryptions functions use the RSA algorithm using the Bouncycastle library [53].

The implementation is split into two key parts, that is, the client side and the server side. Furthermore, it is important the communications between each party which is developed in XML format. Next we briefly depict each side of the protocol.

- **Client side**. As we said, the user application ($\mathcal{U}$) is developed over the Android operating system, using the API level 7 to accomplish compatibility for the two smartphones.
- **Server side**. This side compromises the *Group TTP* ($\mathcal{M}_{\mathcal{G}}$), the *Payment TTP* ($\mathcal{M}_{\mathcal{C}}$), the *Source Station* ($\mathcal{P}_{\mathcal{S}}$) and the *Destination Station* ($\mathcal{P}_{\mathcal{D}}$) servers. Each entity is developed over Java JDK 6.0 and all of them has his own MySQL database to store the useful information. Each entity exposes his service through a server TCP port.
- **Communications**. The messages exchanged by the entities of the AFC infrastructure is serialized with XML and they are sent over a network communication. The XML format has a textual representation, it is portable and it is multiplatform, so it is suitable for our service.

D.8.1 Test scenario

The test scenario we have used is composed by a notebook where are located the above servers while the client side is tested over two Android smartphones. The first one is the HTC Desire and the other one is the HTC Wildfire. The former is a medium-high class device and the latter is a low-medium class smartphone. The mobile phones and notebook features are listed in the Table 11. Testing the application over two types of smartphones can provide us valuable information about the protocol performance over two phone types with different computing capabilities. It will be also interesting because we can predict the future evolution of mobile devices and the expected behavior improvement using new and powerful terminals. The test with each device and each protocol version have been done 20 times in order to get the average time for each protocol step. Finally, the connectivity between the Android client and the AFC is pro-

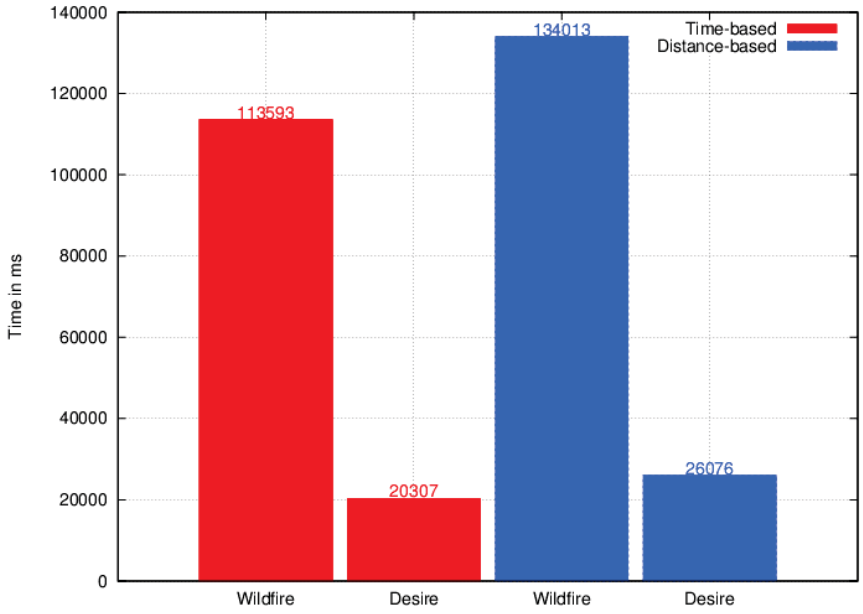


Figure 16: Total time expended for each protocol version over each smartphone.

vided by a wireless 802.11g network using Java Sockets but the connections between each infrastructure server are done over the laptop localhost network interface.

D.8.2 Discussion

After the protocol implementation we now analyze and discuss the time results for each test. We will depict the benchmarks results from less to more detail. So, the first graph in Fig. 16 shows us that the protocol execution over the HTC Desire is faster than over the HTC Wildfire, as expected. If we analyze each protocol version, for the time-based protocol the Wildfire spends around 113 seconds to execute the whole protocol, while Desire spends only 20 seconds, that is, 5.6 times less. Then, for the distance-based protocol the trend remains the same because over Wildfire the modified protocol completes after 134 seconds while the Desire only needs 26 seconds, that is, around 5.2 times less. At first glance, we can say that the difference between both protocols is not as large as we could expect, because over the Desire only expends 6 more seconds to complete while on the Wildfire 21 seconds more.

Next we are going to show a detailed study of the spent time and how we can improve the application performance for both smartphones. So, we will see the spent time in each protocol step to analyze where the application requires more computation power. In order to understand Fig. 17 and Fig. 18, we need to explain what means each step in the x-axis:

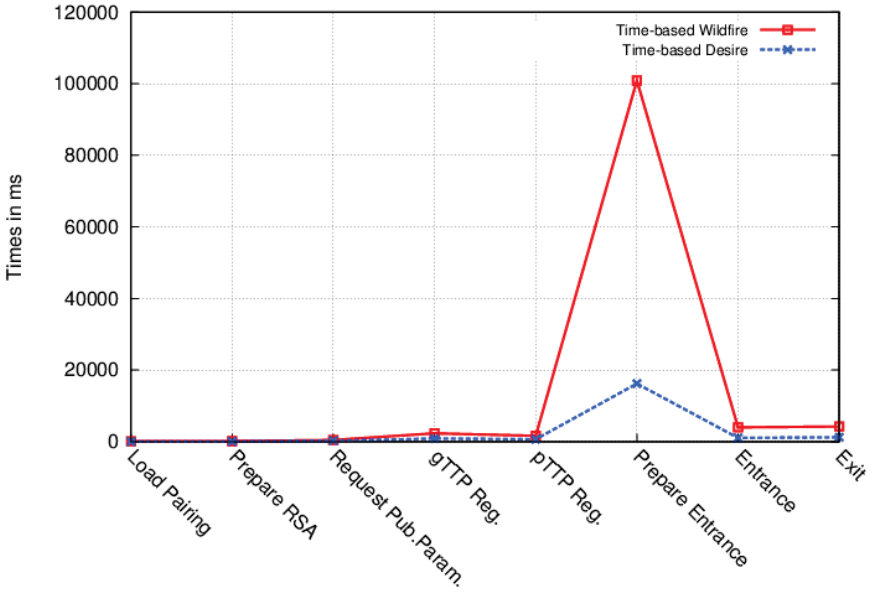


Figure 17: Time flow for the time-based protocol and smartphone

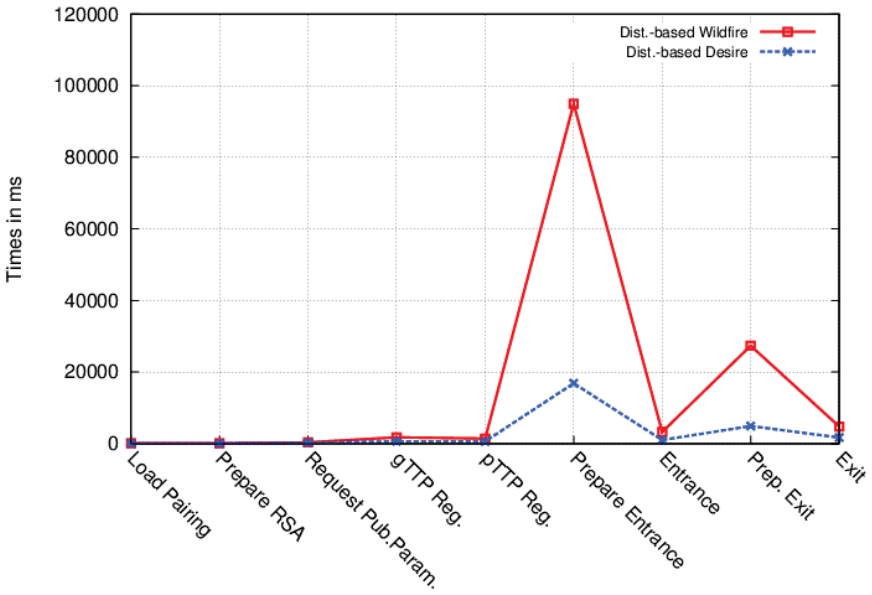


Figure 18: Time flow for the distance-based protocol and smartphone

- **Load Pairing**. Here the user application loads all the data needed to do pairing operations, e.g. loading elliptic curve parameters from a file or preparing some Java objects.
- **Prepare RSA**. At this time the RSA key pair is loaded from a PEM file stored in the phone data store, e.g. from the SD storage.
- **Request Pub.Param**. This is the time needed for the application to request to the Group TTP ($\mathcal{M}_{\mathcal{G}}$) server the public parameter α needed in the next step.
- **gTTP Reg**. This is the time used by the application to complete the registration to the Group TTP ($\mathcal{M}_{\mathcal{G}}$) server. It matches the protocol steps **generatePseudonym** and **keyIssue** from §D.5.3.
- **pTTP Reg**. This time belongs the user registration to the Payment TTP ($\mathcal{M}_{C}$) server. It matches the protocol steps **startingZKP**, **challengeGeneration**, **proofGeneration** and **verifyPseudonym** from §D.5.3.
- **Prepare Entrance**. It is the total precomputation time spent before the system entrance. It applies for both time-based and distance-based protocols. Here all the precomputable parameters needed to build the group signature are established.
- **Entrance**. It is the time spent to enter the system. It matches the protocol step from §D.5.3.
- **Prepare Exit**. It is the time spent by the exit group signature precomputation before the exit step. It applies only for distance-based protocol version.
- **Exit**. Finally, it is the elapsed time by the system exit. It matches the protocol step from §D.5.3.

Then, Fig. 17 shows us the protocol time flow step by step for the time-based protocol. If we analyze the graph, we can see the time is mostly consumed to compute the system entrance step. Therefore, the Desire device is only clearly faster on the system entrance. So, if we use precomputation before the system entrance, the performance of the protocol is similar in both devices and is only clearly greater where more computation power is needed.

Next, Fig. 18 depicts similar data as Fig. 17 but now for the distance-based protocol. The trend is the same as Fig. 17 till the system entrance because previous steps are untouched. After it and before the system exit, appears a new prepare exit step where the client does more precomputations before he arrives to the exit. The important result is that the exit step is done by Desire in around 1.7 seconds while Wildfire in 4.7 seconds, that is, in this case is only 3 seconds more for the less powerful device.

If we cross compare both protocols through each smartphone, we can see that the time needed to execute the modified version, without taking care of the precomputation time which can be computed offline by the client, is not much higher than the time spent by the original version. So we can state that the modified version increases the time cost but it remains usable for both devices.

Fig. 19 and Fig. 20 will expose the different execution times of each protocol stage over both smartphones if we use precomputation or not. Now, in the x-axis we depict the protocol phases as following:

- **Init**. This phase belongs the client application deployment time and also the time needed to request the public parameter α to the *Group TTP*.
- **Registration**. This phase adds the *Group TTP* registration and the *Payment TTP* registration times. So this is the time needed for a user to complete the system registration in order to use it.
- **Entrance**. It is the time required for the user till he receives the entrance ticket.
- **Exit**. It is the time needed for the user to leave the system till he obtains the exit ticket.
- **Pre. Comp**. This is the whole precomputation time. In the time-based protocol the precomputation in only needed before the entrance step while in the distance-based protocol, the precomputation is done both before entrance and exit. Note that this stage only affects Fig. 20.

On the one hand, Fig. 19 shows us that in the time-based protocol, the entrance is the bottleneck of the system. In the distance-based protocol, the entrance step consumes again a lot of time but less than the time-based protocol because an encryption has been erased. Moreover, as a result of the security improvement in the system exit step, is clear that this stage tooks longer to execute because more computation is needed.

On the other hand, in Fig. 20 all the precomputation is taken out of the protocol and stored in the last bar. If we compare this graph set with the previous one, we can see clearly how the precomputations spent most of the time. For example, for the Wildfire smartphone, the entrance tooks around 100 seconds if precomputation is not applied. Instead, if precomputations are activated, the time spent is reduced to around 4.0 seconds. The same happens if we analyze the exit step using Wildfire and the distance-based protocol, because this step tooks around 32 seconds but if precomputations are applied, the time is reduced to only 4.7 seconds.

So, we can conclude the current analysis remarking that the depicted scheme and the implementation developed is suitable to use on AFC systems. Furthermore, the differences between time-based and distance-based protocols execution times are not large. So if we want a higher level of security but with a small time penalty, we can apply the distance-based protocol. Then if we want higher efficiency we can choose the time-based protocol. In both cases, as we can see, the protocol is also suitable for the use with a medium class mobile device as well works faster with a high class smartphone. This suggest that in the near future, when even more powerful mobile devices appear in the market, the protocol performance will be even better.

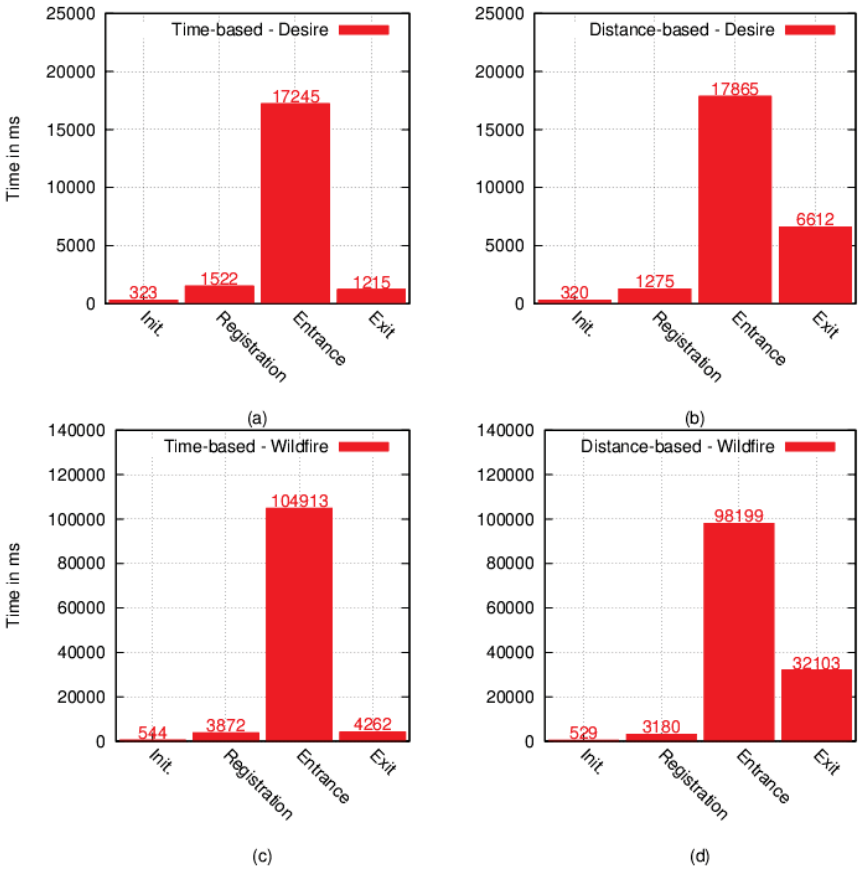


Figure 19: Protocol phases for each protocol version and smartphone.

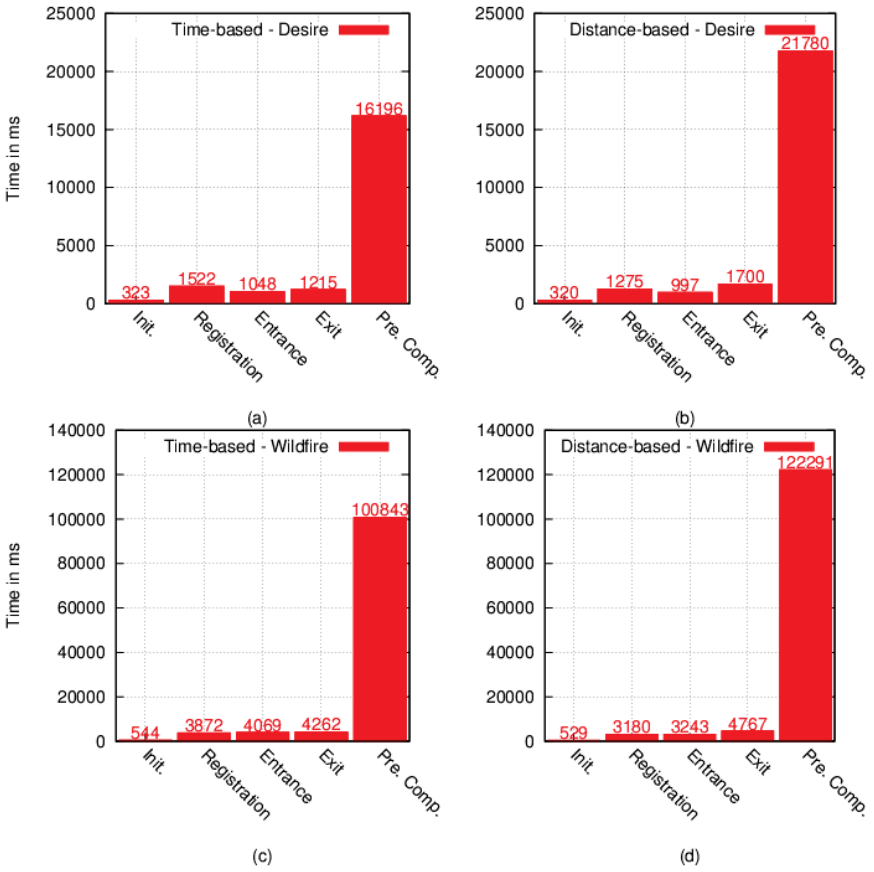


Figure 20: Protocol phases for each protocol version and smartphone.

References

[1] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *Information Theory, IEEE Transactions on*, 44(5):1897 –1905, September 1998.

[2] I.J. Cox, J. Kilian, F.T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. *Image Processing, IEEE Transactions on*, 6(12):1673–1687, Dec 1997.

[3] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2007.

[4] Alexander Barg, G. R. Blakley, and Gregory A. Kabatiansky. Digital fingerprinting codes: problem statements, constructions, identification of traitors. *IEEE Transactions on Information Theory*, 49(4):852–865, 2003.

[5] M. Fernandez and Miguel Soriano. Fingerprinting concatenated codes with efficient identification. In *ISC '02: Proceedings of the 5th International Conference on Information Security*, pages 459–470, London, UK, 2002. Springer-Verlag.

[6] Gábor Tardos. Optimal probabilistic fingerprint codes. *J. ACM*, 55(2):1–24, 2008.

[7] Mark A. Pinsky. *Introduction to Fourier Analysis and Wavelets (Brooks/Cole Series in Advanced Mathematics)*. Thomson Brooks/Cole, 2001.

[8] A. Malpani S. Galperin C. Adams M. Myers, R. Ankney. X.509 internet public key infrastructure - online certificate status protocol - ocsp. June 1999.

[9] Gopal Kakivaya Andrew Layman Noah Mendelsohn Henrik Frystyk Nielsen Satish Thatte Dave Winer Don Box, David Ehnebuske. Simple object access protocol (soap) 1.1. May 2000.

[10] IBM Yin Leng Husband HP Alan Karp HP Keisuke Kibakura Fujitsu Jeff Lancelle Verisign Sam Lee Oracle Sean MacRoibeaird Sun Barbara McKee IBM Tammy Nordan HP Dan Rogers Microsoft Christine Tomlinson Sun Cafer Tosun SAP Tom Bellwood, IBM David Ehnebuske. Uddi version 2.03 data structure reference, July 2002.

[11] Greg Meredith Sanjiva Weerawarana Erik Christensen, Francisco Curbera. Web services description language (wsdl) 1.1, March 2001.

[12] J. Domingo-Ferrer and J. Herrera-Joancomarti. Simple collusion-secure fingerprinting schemes for images. *Information Technology: Coding and Computing, 2000. Proceedings. International Conference on*, pages 128–132, 2000.

[13] Spanair. Spanair y vodafone españa presentan la tarjeta de embarque móvil, 2007. http://www.spanair.com/web/es-es/Sobre-Spanair/Noticias-y-eventos/Spanair-y-Vodafone-Espana-presentan-la-tarjeta-de-embarque-movil/.

[14] IATA. Industry bids farewell to paper ticket, 2008. http://www.iata.org/pressroom/pr/2008-31-05-01.htm.

[15] AMSBUS, 2008. http://www.svt.cz/en/amsbus/.

[16] LeedsUnited. Official leeds sms, 2007. http://www.leedsunited.com/page/Welcome.

[17] Arnau Vives-Guasch, Magdalena Payeras-Capella, Macià Mut-Puigserver, and Jordi Castellà-Roca. E-ticketing scheme for mobile devices with exculpability. In *Data Privacy Management (DPM), Fifth International Workshop*, volume 6514 of *Lecture Notes in Computer Science*, page (to appear), 2010. ISSN 0302-9743.

[18] W.I. Siu and Z.S. Guo. Application of electronic ticket to online trading with smart card technology. *Proceedings of the 6th INFORMS Conference on Information Systems and Technology (CIST-2001)*, pages 222–239, 2001. Miami Beach, Florida (US).

[19] J. Elliot. The one-card trick multi-application smart card e-commerce prototypes. *Computing & Control Engineering Journal*, 10(3):121–128, 1999. IET.

[20] D. Haneberg. Electronic ticketing: risks in e-commerce applications. *Digital excellence*, pages 55–66, 2008. Springer-Verlag, ISBN 3540726209.

[21] Kimio Kuramitsu, Tadashi Murakami, Hajime Matsuda, and Ken Sakamura. Ttp: Secure acid transfer protocol for electronic ticket between personal tamper-proof devices. In *24th Annual International Computer Software and Applications Conference (COMPSAC2000)*, pages 87–92, Taipei, Taiwan, Oct 2000. vol. 24.

[22] K. Kuramitsu and K. Sakamura. Electronic tickets on contactless smartcard database. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 392–402, 2002. LNCS 2453.

[23] S. Matsuo and W. Ogata. Electronic ticket scheme for its. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E86A(1):142–150, 2003.

[24] I.W. Siu and Z.S. Guo. The secure communication protocol for electronic ticket management system. In *8th Asia-Pacific Software Engineering Conference (APSEC2001)*. University of Macau, 2001.

[25] Gerhard Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia. A practical attack on the mifare classic. In *CARDIS '08: Proceedings of the 8th IFIP WG 8.8/11.2 international conference on Smart Card Research and Advanced Applications*, pages 267–282, Berlin, Heidelberg, 2008. Springer-Verlag.

[26] B. Patel and J. Crowcroft. Ticket based service access for the mobile user. *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'97)*, pages 223–233, 1997. Budapest, Hungary.

[27] K. Fujimura, H. Kuno, M. Terada, K. Matsuyama, Y. Mizuno, and J. Sekine. Digital-ticket-controlled digital ticket circulation. *8th USENIX Security Symposium*, pages 229–240, 1999. USENIX.

[28] F. Bao. A scheme of digital ticket for personal trusted device. *15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC04)*, 4:3065–3069, 2004. IEEE.

[29] Dominik Haneberg, Kurt Stenzel, and Wolfgang Reif. Electronic-onboard-ticketing: Software challenges of an state-of-the-art m-commerce application. In K.Pousttchi and K.Turowski, editors, *Workshop Mobile Commerce*, volume 42 of *Lecture Notes in Informatics (LNI)*, pages 103–113. Gesellschaft für Informatik (GI), 2004.

[30] D. Quercia and S. Hailes. Motet: Mobile transactions using electronic tickets. In *1st International Conference on Security and Privacy for Emerging Areas in Communications Networks, Proceedings*, pages 374–383, Athens, Greece, Sep 2005. vol. 24.

[31] Thomas S. Heydt-Benjamin, Hee-Jin Chae, Benessa Defend, and Kevin Fu. Privacy for public transportation. In *6th Workshop on Privacy Enhancing Technologies (PET 2006)*, pages 1–19, 2006. LNCS 4258.

[32] Yu-Yi Chen, Chin-Ling Chen, and Jinn-Ke Jan. A mobile ticket system based on personal trusted device. *Wireless Personal Communications: An International Journal*, 40(4):569–578, 2007.

[33] David Chaum. Blind signatures for untraceable payments. *Advances in Cryptology - CRYPTO'82*, pages 199–203, 1983.

[34] K. Fujimura and Y. Nakajima. General-purpose digital ticket framework. *3rd USENIX Workshop on Electronic Commerce*, pages 177–186, 1998. USENIX.

[35] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[36] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[37] Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25:1606–1621, 2002.

[38] Sarah Clark. All new nokia smartphones to come with nfc from 2011. Near Field Communications World, 2010.

[39] Chad Catacchio. Google's schmidt: Android gingerbread to have near-field-communication support. The Next Web, 2010. http://thenextweb.com/google/2010/11/15/googles-schmidt-android-gingerbread-to-have-near-field-communication-support/.

[40] Jonny Evans. All new nokia smartphones to come with nfc from 2011. Computer World, 2010. http://blogs.computerworld.com/16749/-new_apple_hire_shows_iphone_iwallet_future.

[41] National Institute of Standards and Technology. Special publication 800-57 part 1, 2007. Recommendation for Key Management.

[42] Hua Wang, Jinli Cao, and Yanchuan Zhang. Ticket-based service access scheme for mobile users. *Aust. Comput. Sci. Commun.*, 24(1):285–292, 2002.

[43] Levente Buttyn, Tamas Holczer, and Istvn Vajda. Providing location privacy in automated fare collection systems. In *In Proceedings of the 15th IST Mobile and Wireless Communication Summit, Mykonos, Greece*, June 2006.

[44] Seng-Phil Hong and Sungmin Kang. Ensuring privacy in smartcard-based payment systems: A case study of public metro transit systems. In *Communications and Multimedia Security*, pages 206–215, 2006.

[45] O. Jorns, O. Jung, and G. Quirchmayr. A privacy enhancing service architecture for ticket-based mobile applications. In *2nd International Conference on Availability, Reliability and Security*, pages 374–383, Vienna, Austria, Apr 2007. ARES 2007 - The International Dependability Conference. vol. 24.

[46] Gerald Madlmayr, Peter Kleebauer, Josef Langer, and Josef Scharinger. Secure communication between web browsers and nfc targets by the example of an e-ticketing system. In *EC-Web '08: Proceedings of the 9th international conference on E-Commerce and Web Technologies*, pages 1–10, Berlin, Heidelberg, 2008. Springer-Verlag.

[47] A. Vives-Guasch, J. Castellà-Roca, M. Payeras-Capella, and M. Mut. An electronic and secure automatic fare collection system with revocable anonymity for users. In *8th International Conference on Advances in Mobile Computing & Multimedia (MoMM)*, 2010.

[48] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.

[49] Gartner. Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent. `http://www.gartner.com/it/page.jsp?id=1466313`, November 2010.

[50] IDC. IDC Corporate USA: Android drives big smartphone growth in 2010. `http://www.computerworld.com/s/article/9208478/ Android_drives_big_smartphone_growth_in_2010_IDC_says`, February 2011.

[51] Angelo De Caro. jPBC: Java Pairing-Based Cryptography Library. `http://gas.dia.unisa.it/projects/jpbc/`.

[52] Ben Lynn and et. al. PBC: Pairing-Based Cryptography Library. `http://crypto.stanford.edu/pbc/`.

[53] Bouncy Castle. Bouncy Castle Crypto APIs. `http://www.bouncycastle.org/`.

Table 2: Details of the proposal: security requirements, actors, ticket information and receipt information

Security Requirements			
Authenticity		Non-repudiation	
Integrity		Revocable anonymity	
Non-overspending		Offline verification	
Expiry date		Exculpability	
Reusability			
Actors			
User	$\mathcal{U}$	Service Provider	$\mathcal{P}$
Ticket Issuer	$\mathcal{I}$	Trusted Third Party	$\mathcal{T}$
Ticket Information (T)			
Serial number	Sn	Issuer	Is
Service	Sv	Terms and conditions	Tc
User pseudonym	$\mathsf{Pseu}_{\mathcal{U}}$	Attributes	At
Type of ticket	Ty	Verification data	$\delta_{\mathcal{T},\mathcal{P}}$
Validity time	Tv	Date of issue	Ti
Exculpability ($\mathcal{U}$)	$\mathsf{h}_{(\mathsf{r}_{\mathcal{U}},n)}$	Exculpability ($\mathcal{P}$)	$\mathsf{h}_{(\mathsf{r}_{\mathcal{I}},n)}$
Digital signature of $\mathcal{I}$	$\mathsf{Sign}_{\mathcal{I}}(\mathsf{T})$		
Receipt Information (R)			
Exculpability ($\mathcal{P}$)	$\mathsf{A}_{\mathcal{P}}$	Timestamp	τ_i
Ticket serial number	T.Sn	Digital signature of $\mathcal{P}$	$\mathsf{Sign}_{\mathcal{P}}(\mathsf{R})$

Table 3: Equipment specification details

Computer(server)	CPU	AMD Athlon 64 X2 Dual 5000+ (2.59GHz)
	RAM	2 GB
	OS	Windows XP
	Java version	Java 6
	NFC reader	Arygon ADRA-USB
Mobile phone(client)	Model	Nokia 6212 NFC classic
	Java version	J2ME (Series 40 SDK 1.0 with JSR 257 extension)

Table 4: Details of the pseudonym renewal partial times

Partial time	Description
t_1	Sending of the pseudonym request
t_2	Reception of the pseudonym response
t_3	Decryption of the signed pseudonym
t_4	Pseudonym's signature verification

Table 5: Details of the ticket purchase partial times

Partial time	Description
t_1	Sending of the commitment
t_2	Reception of the challenge
t_3	Computation of the Schnorr's ZKP response
t_4	Sending of the Schnorr's ZKP response
t_5	Computation of the shared symmetric key
t_6	Reception of the ticket
t_7	Verification of the ticket data

Table 6: Details of the ticket verification partial times

Partial time	Description
t_1	Sending of the ticket
t_2	Reception of the response
t_3	Verification of the response
t_4	Computation of the symmetric encryption of $\mathsf{r}_{\mathcal{U}}$
t_5	Sending of the symmetric encryption of $\mathsf{r}_{\mathcal{U}}$
t_6	Reception of the receipt
t_7	Verification of the receipt data
t_8	Computation and verification of the symmetric decryption of $\mathsf{r}_{\mathcal{I}}$

Ref.	Anonymity	Untraceability	Device
[42]	Revocable	No	Mobile
[43]	Revocable	No	Smart-card
[31]	Revocable	Yes	Mobile and Smart-card
[44]	Revocable	No	Smart-card
[45]	Revocable	No	Mobile
[46]	Revocable	No	Mobile

Table 7: Comparison of the analyzed proposals

Entrance ticket ($t_{in}{}^*$)		
Name	Notation	Description
Serial number	Sn	generated by $\mathcal{P}_S$
Entrance station	Ps	$\mathcal{P}_S$ identifier
Entrance timestamp	τ_1	time entry system
$\mathcal{U}$'s commitment	σ^*	signed by $\mathcal{U}$
Digital signature	$Sign_{\mathcal{P}_S}(t_{in})$	content signed by $\mathcal{P}_S$

Table 8: Information in entrance ticket

Exit ticket ($t_{out}{}^*$)		
Name	Notation	Description
t_{in}'s serial number	t_{in}.Sn	sent by $\mathcal{U}$
Destination station	Pd	
Paid fare	a	
Payment timestamp	τ_2	time exit ticket
Digital signature	$Sign_{\mathcal{P}_\mathcal{D}}(t_{out})$	content signed by $\mathcal{P}_\mathcal{D}$

Table 9: Information in exit ticket

Notation information			
Name	Notation	Name	Notation
Group public key	gpk	List of group private keys	$gsk[\,]$
List of group revocations	$grt[\,]$	Exponentiation base	α
Prime number	p	Prime number	q
$\mathcal{U}$'s pseudonym (for payment)	$y_\mathcal{U}$	Inverse exponentiation of $y_\mathcal{U}$ (secret)	$x_\mathcal{U}$
j-th random number	r_j	Exponentiation of r_j	s_j
j-th challenge for $\mathcal{U}$ to show authorship of $y_\mathcal{U}$	c_j	Challenge c_j's response by $\mathcal{U}$	ω_j
Probabilistic encryption of $y_\mathcal{U}$	$\delta_\mathcal{U}$	j-th timestamp	τ_j
Verification parameter	k	Hash image of parameter k	h_k
Digital signature of the content c generated by the entity E	$Sign_E(c)$	$\mathcal{U}$'s commitment	σ^*
Entrance ticket, signed by $\mathcal{P}_S$	$t_{in}{}^*$	t_{in} serial number	Sn
Source service provider identifier	Ps	Exit ticket, signed by $\mathcal{P}_\mathcal{D}$	$t_{out}{}^*$
Challenge & fare, signed by $\mathcal{P}_\mathcal{D}$ for $\mathcal{U}$	β^*	Fare calculation function	$f()$
Fare to be paid	a	Destination service provider identifier	Pd
Probabilistic encryption of $\mathcal{U}$'s verification data	$\gamma_\mathcal{U}$	Probabilistic encryption of $\mathcal{P}_\mathcal{D}$'s verification data	$\gamma_{\mathcal{P}_\mathcal{D}}$
Payment acceptance signed by $\mathcal{M}_C$	ok^*	Payment rejection signed by $\mathcal{M}_C$	ko^*

Table 10: Notation information, in appearance order

Device	CPU	RAM	ROM	OS
Notebook	Intel Core Duo 2 1.6GHz	4GB		Debian Linux 5.0
HTC Desire	Qualcomm Snapdragon 1GHz	576MB	512MB	Android 2.2
HTC Wildfire	Qualcomm MSM7225 528MHz	384MB	512MB	Android 2.1

Table 11: Technical features of test devices.