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

Andreu Pere Isern-Deyà[1], Arnau Vives-Guasch[2], Macià Mut-Puigserver[1], Magdalena Payeras-Capellà[1] and Jordi Castellà-Roca[2]

[1]*Dpt. de Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears, Ctra. de Valldemossa, km 7,5. 07120 Palma de Mallorca, Spain*
[2]*Dpt. d'Enginyeria Informàtica i Matemàtiques, UNESCO Chair in Data Privacy, Universitat Rovira i Virgili, Av. Països Catalans 26, E-43007 Tarragona, Catalonia, Spain*
Email: {andreupere.isern, macia.mut, mpayeras}@uib.es, {arnau.vives, jordi.castella}@urv.cat

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 helps 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 journeys of the same user are not linkable between them. 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 be used 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.

Keywords: Security; Privacy; Mobile Applications; E-commerce; Applied Cryptography

1. INTRODUCTION

The incorporation of Information and Communication Technologies (ICT) in Automatic Fare Collection (AFC) systems allows to reduce costs and 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 transport. 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 that, the design of efficient and secure AFC systems becomes necessary. 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 user profiles. 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 escape from the security forces. Thus, we have two opposed requirements. In order to fulfill both requirements, the system could revoke anonymity of a determined user by means of a court order. Another consideration is the high number of users in an AFC system, requiring then the system to be very fast in both entrance and exit operations.

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, as the system offers revocable anonymity. Moreover, users do not need to obtain a new credential every time that they join the AFC system. This feature improves the system usability, because in the previous AFC systems new credential information is needed for every new journey.

2. STATE-OF-THE-ART

We have performed an analysis of AFC proposals that consider anonymity for users, offering revocable anonymity [1, 2, 3, 4, 5, 6].

In the majority of these schemes, the provider can link different journeys from the same user [1, 2, 4, 5, 6]. 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 users' profiles. These profiles are useful for the provider because they can be used generically to improve the transportation system or more concretely to define a commercial product specifically for one profile. Nonetheless, the creation of users' profiles is a serious violation of the privacy, and the AFC system must avoid the tracking of the users.

In [3] 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.) [1, 3, 5, 6] for these systems, instead of smart cards [2, 3, 4]. Thus, we can say that the mobile device is a user's requirement in the AFC systems.

In Table 1 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 [3], where the credential renewal means an important extra cost.

Ref.	Anonymity	Untraceability	Device
[1]	Revocable	No	Mobile
[2]	Revocable	No	Smart-card
[3]	Revocable	Yes	Mobile and Smart-card
[4]	Revocable	No	Smart-card
[5]	Revocable	No	Mobile
[6]	Revocable	No	Mobile

TABLE 1. Comparison of the analyzed proposals

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.

2.1. 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. This concept can be extended to 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 the period of time that the service has been used. Thus, in this case, a proper timestamp has to be generated when the user gets in and out of the system. The difference between the present time and the initial timestamp will determine the fare to be paid to the service provider. 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).

2.2. Distance-Based Systems

Distance-based AFC systems are more common than Time-based systems. Some examples of Distance-Based Collection Systems are tolls and public transport services such as subway or 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 §6.3.

2.3. 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 [7], 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 to put system exits 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 time-based 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 3, 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 4. Note that [7] does not include the requirements for the above extensions. The time-based protocol is described in Section 5 and the distance-based in Section 6. The Section 7 contains the security analysis of both protocols. The protocols' implementation is depicted in Section 8. The protocols have been implemented in the Android mobile platform and its performance has been evaluated in two Android smartphones. Finally, the conclusions are presented in Section 9.

3. BACKGROUND

We use the short group signature (BBS) scheme [8] 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. The algorithm then has the following steps: Select $h \overset{R}{\leftarrow} G_1 \backslash \{1_{G_1}\}$ and $\xi_1, \xi_2 \overset{R}{\leftarrow} \mathbb{Z}_p^*$, and set $u, v \in G_1$ such that $u^{\xi_1} = v^{\xi_2} = h$. Select $\gamma \overset{R}{\leftarrow} \mathbb{Z}_p^*$ 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 $x_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$ 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 $\alpha, \beta \overset{R}{\leftarrow} \mathbb{Z}_p$ 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 $r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2} \overset{R}{\leftarrow} \mathbb{Z}_p$ 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:
  $$c \overset{?}{=} H(M, T_1, T_2, T_3, \tilde{R_1}, \tilde{R_2}, \tilde{R_3}, \tilde{R_4}, \tilde{R_5})$$

- $Open_G(gpk, gmsk, M, \sigma)$. 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.

3.1. Linkability between signatures

In the system, all the users have to be anonymous, and also their signatures have to be unlinkable one each other. However, for security, and only in some cases, determined signatures of a same user could be linkable between them.

3.1.1. 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 $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$;
  - 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.

3.1.2. 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):

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

4. REQUIREMENTS OF THE FARE COLLECTION SYSTEMS

4.1. 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.

4.2. 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
- 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

4.3. Requirements for Distance-Based Systems

Distance-based systems, as described in §2.2 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.
- Make a correlation between the distance covered by the user and the fair to be paid. The larger is the distance, the higher is the fare.

In distance-based systems, beside the entrance point, the entrance ticket must include a new item, which is the direction of the journey 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 appeared problems due to the possibility of confabulated attacks. The solution for compliant services is described in §6.1 while the solution for non-compliant services is included in §6.4.

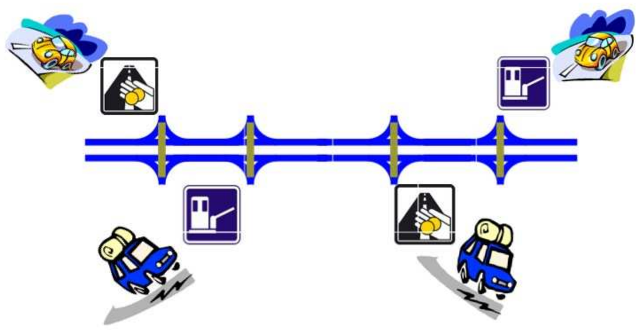

FIGURE 1. Separated directions in a compliant distance-based AFC.

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 [8] 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.

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.

5.2. Ticket information

In this section, we describe the information that is included in the entrance ticket at Table 2 and the information in the exit ticket at Table 3. To give a description of the protocols we use the notation described at Table 4.

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 $\mathcal{P_S}$	$\mathsf{t_{in}}^*$	$\mathsf{t_{in}}$ serial number	Sn
Source service provider identifier	Ps	Exit ticket, signed by $\mathcal{P_D}$	$\mathsf{t_{out}}^*$
Challenge & fare, signed by $\mathcal{P_D}$ 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 $\mathcal{P_D}$'s verification data	$\gamma_{\mathcal{P_D}}$
Payment acceptance signed by $\mathcal{M_C}$	ok^*	Payment rejection signed by $\mathcal{M_C}$	ko^*

TABLE 4. Notation information, in appearance order

Entrance ticket ($\mathsf{t_{in}}^*$)		
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	$Sign_{\mathcal{P_S}}(\mathsf{t_{in}})$	content signed by $\mathcal{P_S}$

TABLE 2. Information in entrance ticket

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

TABLE 3. Information in exit ticket

5.3. Protocol Specification

5.3.1. Phases

In the protocol, there are the following phases:

- Setup: $\mathcal{M_G}$ generates all the group keys, revocation lists, etc.
- User Registration: $\mathcal{U}$ registers at $\mathcal{M_G}$, she receives a group key pair. $\mathcal{U}$ also registers at $\mathcal{M_C}$ through a pseudonym that will be used only for payments. In the AFC system, $\mathcal{M_C}$ 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 $\mathcal{P_S}$, she receives an entrance ticket from $\mathcal{P_S}$ that will have to be showed in the destination station.
- System exit: the user performs a weak authentication to the destination checkpoint $\mathcal{P_D}$ and shows the entrance ticket. $\mathcal{P_D}$ 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, $\mathcal{P_D}$ 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.

5.3.2. Setup

This phase is executed once at first. $\mathcal{M_G}$ 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.

5.3.3. User Registration

$\mathcal{U}$ registers at the group TTP $\mathcal{M_G}$ 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 $\mathcal{M_G}$; 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 [9] proving knowledge of $x_{\mathcal{U}}$ without disclosing that secret. Thus, privacy is preserved for users, but this anonymity could be revoked by $\mathcal{M_G}$ 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 $Sign_{\mathcal{U}}(y_{\mathcal{U}},$ *'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 $\mathcal{M}_{\mathcal{C}}$;

challengeGeneration: $\mathcal{M}_{\mathcal{C}}$ 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 $\mathcal{M}_{\mathcal{C}}$;

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

5.3.4. 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 $r_1 \overset{R}{\leftarrow} \mathbb{Z}_q$;
2. computes $s_1 = \alpha^{r_1} \pmod p$;
3. computes $\delta_{\mathcal{U}} = PK_{\mathcal{M}_{\mathcal{C}}}(y_{\mathcal{U}})$ [3];
4. generates a random value $k \overset{R}{\leftarrow} \mathbb{Z}_q$;
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;

[3]The cryptosystem is probabilistic

5.3.5. 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 $\sigma.h_k \overset{?}{=} hash(k)$, 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): $a = f_t(\mathsf{t_{in}}.\mathsf{Ps}, \mathsf{Pd}, \mathsf{t_{in}}.\tau_1, \tau_2)$; Therefore, in this case, $f_t()$ is a function specially designed for computing the fare between two stations on a time-based fare system.
6. generates a challenge $c_1 \overset{R}{\leftarrow} \mathbb{Z}_q$;
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 $\gamma_{\mathcal{U}} = PK_{\mathcal{M}_{\mathcal{C}}}(\omega_1, \mathsf{t_{in}}.\mathsf{Sn}, \beta.a)$;
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 $\mathcal{M}_{\mathcal{C}}$;

verifyPayment: $\mathcal{M}_{\mathcal{C}}$ 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: $\alpha^{\omega_1} \overset{?}{=} s_1 \cdot (y_{\mathcal{U}})^{c_1}$;
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 $ko^* = (ko, Sign_{\mathcal{M}_{\mathcal{C}}}(ko))$, 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 $\mathcal{P_D}$;

setExit: $\mathcal{P_D}$ computes:

1. composes $\mathsf{t_{out}} = (\mathsf{t_{in}}.\mathsf{Sn}, \mathsf{Pd}, \beta.a,$ 'leave taking at $\beta.\tau_2$') and signs it $\mathsf{t_{out}}^* = (\mathsf{t_{out}}, Sign_{\mathcal{P_D}}(\mathsf{t_{out}}))$;
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.

5.4. User's Claims

During the *System exit* protocol, $\mathcal{P_D}$ could not follow the protocol due to different reasons (i.e. $\mathcal{P_D}$ 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.

5.4.1. Claim 1: an incorrect β^* is received

During the *System exit* protocol, $\mathcal{U}$ can send the validation information $(\mathsf{t_{in}}^*, k)$, but $\mathcal{P_D}$ could misbehave and sends a wrong β^* (e.g. the message has an inaccurate τ_2) to $\mathcal{U}$ or, simply, $\mathcal{P_D}$ 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 $\mathcal{P_S}$;
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. 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 $c_1 \overset{R}{\leftarrow} \mathbb{Z}_q$;
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.

5.4.2. 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 $\mathcal{P_D}$ 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 $\mathcal{P_S}$;
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. 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 $\mathcal{P_D}$ of its misbehavior or communication problems with users. $\mathcal{M_C}$ also has to alert $\mathcal{P_D}$ of possible further actions if this problem persists.

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.

5.5.1. Claim 3: An incorrect $(\mathsf{t_{in}}^*, k)$ is received

During the *System exit* protocol, $\mathcal{P_D}$ 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 $\mathcal{P_D}$ sends $(\mathsf{t_{in}}^*, k)$ to $\mathcal{M_G}$;
appealingUser: The user $\mathcal{U}$ is required to send also $(\mathsf{t_{in}}^*, k)$ 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 $(\mathsf{t_{in}}^*, k)$ which is generated by $\mathcal{P_S}$;
2. verifies the link with the hash value $\mathsf{t_{in}}.\sigma.h_k \stackrel{?}{=} hash(k)$; If the link is not verified, $\mathcal{M_G}$ 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 $\mathcal{P_D}$ and $y_{\mathcal{U}}$ to $\mathcal{M_C}$;
5. $\mathcal{U}_i$ is added to the revocated list;

5.5.2. *Claim 4: An incorrect $\gamma_{\mathcal{U}}$ is received*

During the *System exit* protocol, $\mathcal{P_D}$ and $\mathcal{M_C}$ 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 $\mathcal{M_C}$ composes a payment rejection ko = ('verification information error', $\gamma_{\mathcal{U}}$), signs it $ko^* = (ko, Sign_{\mathcal{M_C}}(ko))$ and sends it to $\mathcal{P_D}$. The Payment TTP $\mathcal{M_C}$ also sends $(sk_{\mathcal{P_D}}(\gamma_{\mathcal{U}}), \gamma_{\mathcal{P_D}})$ to $\mathcal{M_G}$ and stops the protocol.

providerInfo: The destination service provider $\mathcal{P_D}$ sends $(\mathsf{t_{in}}^*, k)$ to $\mathcal{M_G}$;

appealingUser: The user $\mathcal{U}$ is required to send also $(\mathsf{t_{in}}^*, k, \gamma_{\mathcal{U}})$ to $\mathcal{M_G}$, in order to avoid false accusations;

claim4Response: If $\mathcal{U}$ does not send the required items, the Group TTP $\mathcal{M_G}$ computes:

1. verifies if the decrypted information of $\gamma_{\mathcal{U}}$, $\gamma_{\mathcal{P_D}}$ 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 $\mathcal{P_D}$ and $y_{\mathcal{U}}$ to $\mathcal{M_C}$;
4. $\mathcal{U}_i$ is added to the revocated list;

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 [8] for mass-transport services. We have adapted the Time-Based protocol in §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 §4.3.

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

6.1. Requirement Compliant Distance-Based Systems

Little changes have to be made to the protocol presented in §5 in order to adapt it to a distance-based automated fare collection system that fulfils the requirements presented in §4.3. 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 $\mathcal{P_S}$ 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: $\mathsf{t_{in}}^* = (\mathsf{t_{in}}, Sign_{\mathcal{P_S}}(\mathsf{t_{in}}))$;
4. sends $\mathsf{t_{in}}^*$ to $\mathcal{U}$;

Concerning the protocol specifications, we have only to change the action **verifyTicket** performed by $\mathcal{P_D}$ 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 $\mathcal{P_D}$ performs:

1. verifies the signature of $\mathsf{t_{in}}^*$ which is computed by $\mathcal{P_S}$;
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 $c_1 \stackrel{R}{\leftarrow} \mathbb{Z}_q$;
8. composes $\beta = (\mathsf{t_{in}}^*, k, a, c_1, \tau_2, \mathsf{Pd})$, and signs it $\beta^* = (\beta, Sign_{\mathcal{P_D}}(\beta))$;
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 $\gamma_{\mathcal{P_D}} = (\beta.a, \mathsf{t_{in}}.\mathsf{Sn}, \mathsf{t_{in}}.\sigma, c_1)$;

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

claim1Response: The Payment TTP $\mathcal{M}_\mathcal{C}$ 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^*, $\mathcal{M}_\mathcal{C}$ 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 ($\mathsf{t_{in}.Ps}$) and the exit station (Pd):
   $a = f_d(\mathsf{Pd}, \mathsf{t_{in}.Ps}, \mathsf{t_{in}}.\tau_1, \tau_2)$;
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 $\beta^* = (\beta, Sign_{\mathcal{M}_\mathcal{C}}(\beta))$;
9. sends β^* to $\mathcal{U}$;

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

claim2Response: The Payment TTP $\mathcal{M}_\mathcal{C}$ 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 ($\mathsf{t_{in}.Ps}$) and the exit station ($\beta.\mathsf{Pd}$): $a = f_d(\mathsf{t_{in}.Ps}, \beta.\mathsf{Pd}, \mathsf{t_{in}}.\tau_1, \beta.\tau_2)$. Then, $\mathcal{M}_\mathcal{C}$ 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 $\mathsf{t_{out}} = (\mathsf{t_{in}.Sn}, \beta.a,$ 'leave taking at $\beta.\tau_2$'), and signs it $\mathsf{t_{out}}^* = (\mathsf{t_{out}}, Sign_{\mathcal{M}_\mathcal{C}}(\mathsf{t_{out}}))$;
7. sends $\mathsf{t_{out}}^*$ to $\mathcal{U}$;

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 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.

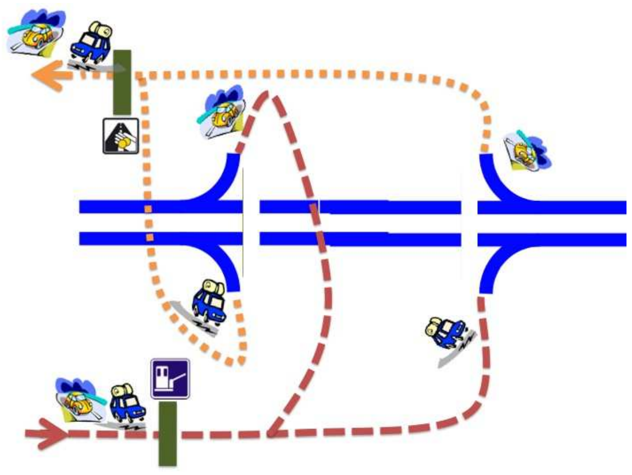

FIGURE 2. Non-compliant distance-based AFC.

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. 2, 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. 3, we have analyzed the fraud ratio, regarding the cases where users could take advantage of this 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.

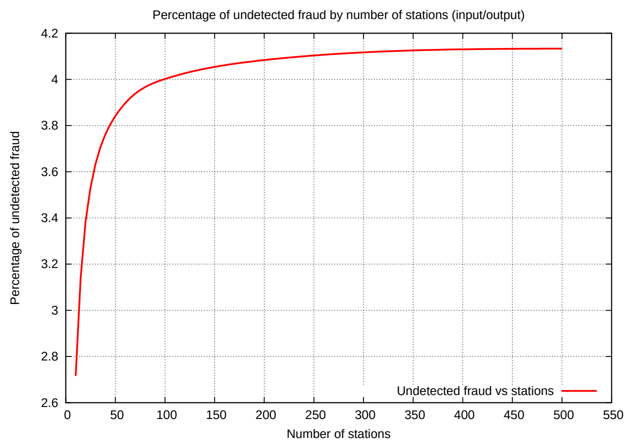

FIGURE 3. Percentage of undetected fraud by number of stations in a linear scenario (in %)

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 §3 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 §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 $\delta_{\mathcal{U}} = PK_{\mathcal{M}_\mathcal{C}}(y_{\mathcal{U}})$ [4];
4. composes $\sigma = (s_1, \delta_{\mathcal{U}})$, and signs it with $gsk[i]$, her private group key: $\sigma^* = (\sigma, \bar{\sigma} = Sign_G(gpk, gsk[i], \sigma))$;
5. 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 $\mathsf{t_{in}} = (\mathsf{Sn}, \mathsf{Ps}, \tau_1, \tau_v, \sigma^*, \xi)$ and signs it: $\mathsf{t_{in}}^* = (\mathsf{t_{in}}, Sign_{\mathcal{P}_\mathcal{S}}(\mathsf{t_{in}}))$;
4. sends $\mathsf{t_{in}}^*$ to $\mathcal{U}$;

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

[4]The cryptosystem is probabilistic

previousStep: $\mathcal{P}_\mathcal{D}$ generates a $\Phi \overset{R}{\leftarrow} \mathbb{Z}_p$ 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 (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 $\beta^* = (\beta, Sign_{\mathcal{P}_\mathcal{D}}(\beta))$;
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 $\mathcal{M}_\mathcal{C}$;

claim1Response: The Payment TTP $\mathcal{M}_\mathcal{C}$ 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^*, $\mathcal{M}_\mathcal{C}$ 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. $\mathcal{M}_\mathcal{C}$ 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 (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 $\beta^* = (\beta, Sign_{\mathcal{M_C}}(\beta))$;
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 (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 $\mathcal{P_D}$ sends $(\mathsf{t_{in}}^*, \Phi^*)$ to $\mathcal{M_G}$;

appealingUser: The user $\mathcal{U}$ is required to send also $(\mathsf{t_{in}}^*, \Phi^*, \gamma_{\mathcal{U}})$ to $\mathcal{M_G}$, in order to avoid false accusations;

claim4Response: If $\mathcal{U}$ does not send the required items, the Group TTP $\mathcal{M_G}$ computes:

1. verifies if the decrypted information of $\gamma_{\mathcal{U}}$, $\gamma_{\mathcal{P_D}}$ 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 $\mathcal{P_D}$ and $y_{\mathcal{U}}$ to $\mathcal{M_C}$;
4. $\mathcal{U}_i$ is added to the revocated list;

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: $\mathsf{t_{in}}^* = (\mathsf{t_{in}}, Sign_{\mathcal{P_S}}(\mathsf{t_{in}}))$ and $\mathsf{t_{out}}^* = (\mathsf{t_{out}}, Sign_{\mathcal{P_D}}(\mathsf{t_{out}}))$, and also the sent information before the payment $\beta^* = (\beta, Sign_{\mathcal{P_D}}(\beta))$. If an unauthorized entity can create a valid ticket (entrance or exit) without knowledge of the private keys of $\mathcal{P_S}$ nor $\mathcal{P_D}$, 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 1. According to the definitions given at §4.1 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 ($\mathcal{P}_\mathcal{S}$ and $\mathcal{P}_\mathcal{D}$) can not access to this information because they need the private key of the TTP. In the system, users compute a group signature ($\mathsf{t_{in}}.\sigma^* = (\sigma, Sign_\mathcal{G}(\sigma))$) 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 $\mathcal{M}_\mathcal{C}$ and the group TTP $\mathcal{M}_\mathcal{G}$. 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 $\mathcal{M}_\mathcal{C}$. Then, $\mathcal{M}_\mathcal{C}$ has knowledge about $y_\mathcal{U}$ from the pair $(x_\mathcal{U}, y_\mathcal{U})$ where $y_\mathcal{U} = \alpha^{x_\mathcal{U}} \pmod p$, which identifies her as a valid user; then, the user authenticates by proving knowledge of $x_\mathcal{U}$ through Schnorr's ZKP [9].

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 [8] 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 2. According to the definitions given at §4.1 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 $\mathcal{M}_\mathcal{G}$ 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 $\mathcal{P}_\mathcal{D}$ 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 $\mathsf{t_{in}}^*$ which is signed by $\mathcal{P}_\mathcal{S}$.

Result of Proposition 3. According to the definitions given at §4.1 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 §5 cannot be attacked by confabulated users.*

Proof. The confabulated attack described in §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 §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 §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. §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 4. According to the attack described in §6 and the protocols described in §5 and §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.

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 [5][6] 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) [7] library. This library is a full Java port of PBC (Pairing Based Cryptography) [8] 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 [9].

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.

[5]http://www.gartner.com/it/page.jsp?id=1466313
[6]http://www.computerworld.com/s/article/9208478/
[7]http://gas.dia.unisa.it/projects/jpbc/
[8]http://crypto.stanford.edu/pbc/
[9]http://www.bouncycastle.org/

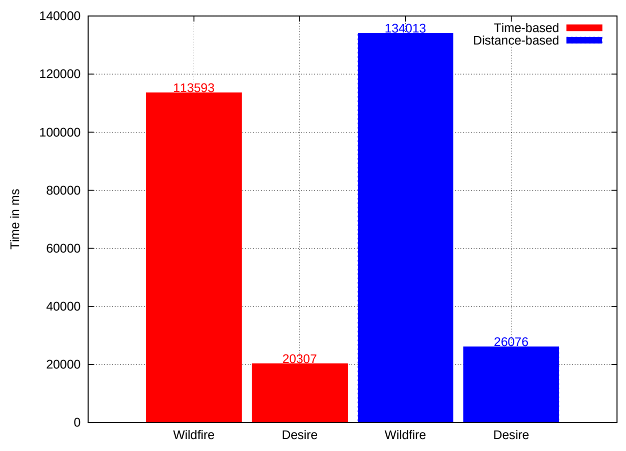

FIGURE 4. Total time expended for each protocol version over each smartphone.

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

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 5. Technical features of test devices.

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. 4 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. 5 and Fig. 6, we need to explain what means each step in the x-axis:

- **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 §5.3.3.
- **pTTP Reg**. This time belongs the user registration to the Payment TTP ($\mathcal{M}_\mathcal{C}$) server. It matches the protocol steps **startingZKP**, **challengeGeneration**, **proofGeneration** and **verifyPseudonym** from §5.3.3.
- **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.

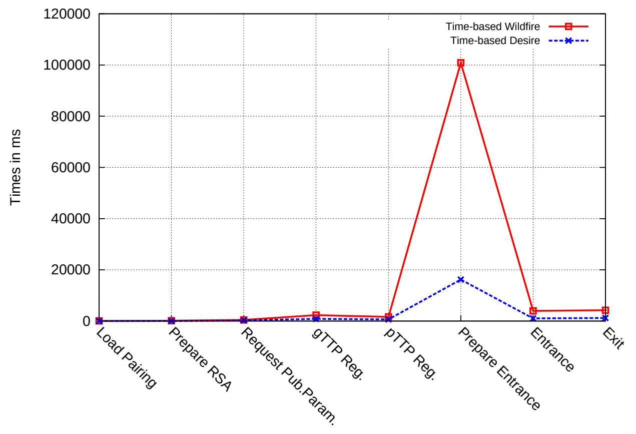

FIGURE 5. Time flow for the time-based protocol and smartphone

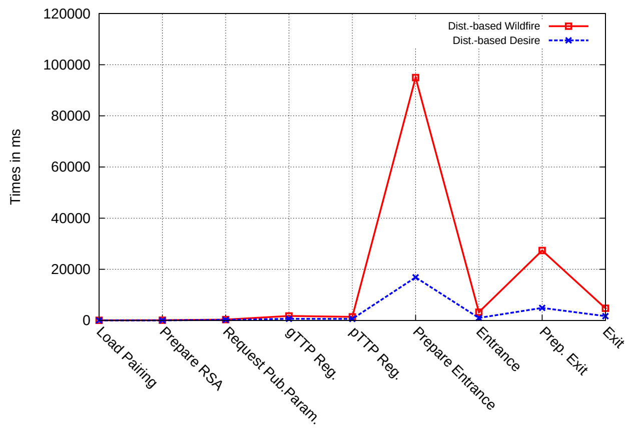

FIGURE 6. Time flow for the distance-based protocol and smartphone

- **Entrance**. It is the time spent to enter the system. It matches the protocol step from §5.3.4.
- **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 §5.3.5.

Then, Fig. 5 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. 6 depicts similar data as Fig. 5 but now for the distance-based protocol. The trend is the same as Fig. 5 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. 7 and Fig. 8 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. 8.

On the one hand, Fig. 7 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. 8 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

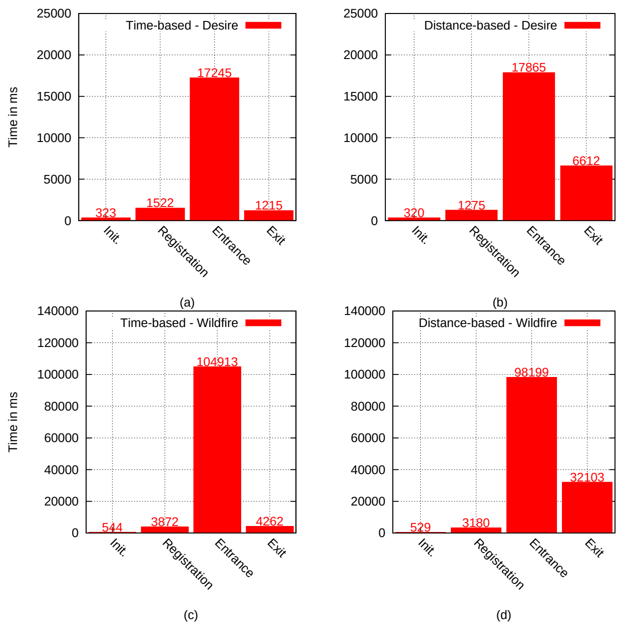

FIGURE 7. Protocol phases for each protocol version and smartphone.

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.

9. CONCLUSIONS

We have presented an electronic and secure automatic fare collection system which is adapted for massive users transport. We have achieve a system that can be adapted to time-based or distance-based fares with slightly changes. The use of group signatures schemes allows the user authentication while it is preserved her privacy. However the identity of users could be disclose in case of a user's misbehaviour. Our system, differently than others, does not require to obtain a new credential every time the user joins in the system in order to obtain untraceability and to prevent tracking and profiling. As a result, the paper proposed a secure automated fare collection resistant to attacks of confabulated user as we have demonstrated in §7.

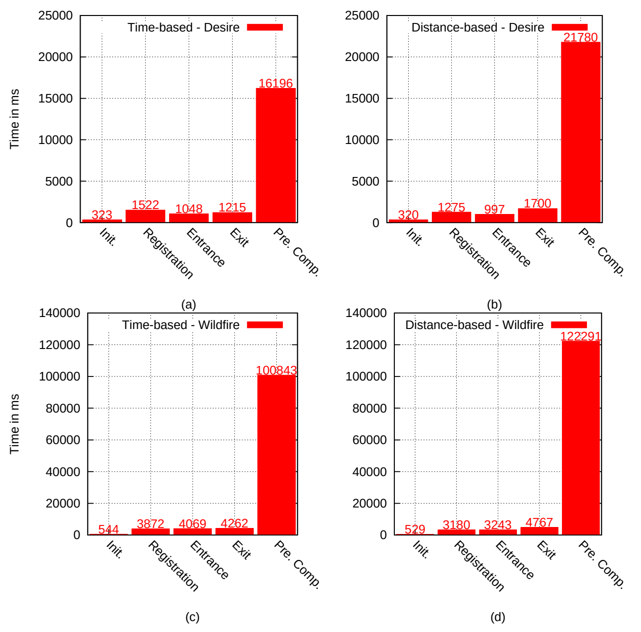

FIGURE 8. Protocol phases for each protocol version and smartphone.

In addition to that, thanks to the implementation of our system we can state the our scheme is realistic and efficient.

The future work goes in the direction to extend our implementation to new communication technologies more suitable for e-commerce procedures such us the near field communication, or NFC, which is intended mainly for use in smartphones.

FUNDING

This work was partially supported by the Spanish Ministry of Science and Innovation [TSI2007-65406-C03-01, ARES-CONSOLIDER INGENIO 2010 CSD2007-00004, "RIPUP" TIN2009-11689, Audit Transparency Voting Process PT-430000-2010-31]; the Spanish Ministry of Industry, Commerce and Tourism ["eVerification2' TSI-020100-2011-39, SeCloud TSI-020302-2010-153]; and the Government of Catalonia [2009 SGR1135]. The authors are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization.

REFERENCES

[1] Wang, H., Cao, J., and Zhang, Y. (2002) Ticket-based service access scheme for mobile users. *Aust. Comput. Sci. Commun.*, **24**, 285–292.

[2] Buttyán, L., Holczer, T., and Vajda, I. (2006) Providing location privacy in automated fare collection systems. *In Proceedings of the 15th IST Mobile and Wireless Communication Summit, Mykonos, Greece*, June.

[3] Heydt-Benjamin, T. S., Chae, H.-J., Defend, B., and Fu, K. (2006) Privacy for public transportation. *6th Workshop on Privacy Enhancing Technologies (PET 2006)*, pp. 1–19. LNCS 4258.

[4] Hong, S.-P. and Kang, S. (2006) Ensuring privacy in smartcard-based payment systems: A case study of public metro transit systems. *Communications and Multimedia Security*, pp. 206–215.

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

[6] Madlmayr, G., Kleebauer, P., Langer, J., and Scharinger, J. (2008) Secure communication between web browsers and nfc targets by the example of an e-ticketing system. *EC-Web '08: Proceedings of the 9th international conference on E-Commerce and Web Technologies*, Berlin, Heidelberg, pp. 1–10. Springer-Verlag.

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

[8] Boneh, D., Boyen, X., and Shacham, H. (2004) Short group signatures. In Franklin, M. K. (ed.), *CRYPTO*, Lecture Notes in Computer Science, **3152**, pp. 41–55. Springer.

[9] Schnorr, C.-P. (1991) Efficient signature generation by smart cards. *J. Cryptology*, **4**, 161–174.