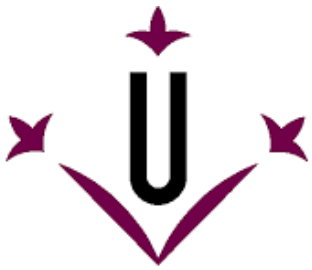

Universitat de Lleida

DOCTORAL THESIS

PRIVACY-PRESERVING PROTOCOLS FOR VEHICULAR TRANSPORT SYSTEMS

Ricard Borges i Llorens

This thesis is presented for the degree of Doctor from the
Universitat de Lleida
Doctoral Program in Engineering and Information Technology

Adviser
Prof. Francesc Sebé i Feixas

2021

RICARD BORGES LLORENS - DNI 43740776C

Firmado digitalmente por RICARD BORGES LLORENS - DNI 43740776C
Fecha: 2021.05.24 10:35:26 +02'00'

Ricard!

Mollerussa, 14th January 2021

Revised 24th May 2021

Abstract

Nowadays, the use of smart devices connected to the Internet by most of the population is a reality. In an *e-society*, most transactions and procedures can be performed digitally. This provides several benefits to the society to which we belong. Nevertheless, new challenges to overcome have appeared.

This thesis is focused on the privacy of citizens while using vehicular transport systems within an *e-society* frame. Specifically, the thesis contributes to two subcategories. The first one refers to pay-by-phone systems for parking vehicles in regulated public areas. The second one is about the use of e-tickets in public transport systems allowing transfers between connecting lines.

Traditionally, payment systems for regulated public parking areas have been based on the use of pay and display machines. The customer purchases a ticket from a machine and displays it on the dashboard of the car. Tickets are usually acquired by inserting coins into the pay station so that the identity of customers is not revealed. Nowadays, several *apps* running on the smartphone of customers are available for making this type of payments. The digitization of this process involves the automatic collection of data about all the parking transactions performed by all the users which could be used to deduce sensitive information about people's mobility.

Public transport systems with transfers between connecting lines are based on the reusability of tickets in a limited way during an established period of time. There are several proposals allowing users to purchase one-use *e-tickets* that guarantee the privacy of users. Current proposals allowing reusability are not adequate when *in-situ* inspections are possible. After an inspection, all the details about user's journey can be linked and deanonymized.

A careful analysis of data collected by service providers can provide sensitive personal information such as: work schedule, profession, hobbies, health problems, political tendencies, sexual inclinations, etc. Although the law, like the European GDPR, requires the correct use of the data collected by service providers, data can be used for illegal purposes after being stolen as a result of a cyber-attack or after being leaked by an internal dishonest employee. Therefore, the design of privacy-preserving solutions for mobility-based services is mandatory in the *e-society*.

Resum

L'ús de dispositius intel·ligents, com els *smartphones*, interconnectats a través d'Internet és una realitat des de fa anys. En una *e-society*, part de l'entorn tradicional es veu reemplaçat per l'entorn digital. Aquest canvi implica millores i beneficis per a les societats de les quals formem part però també apareixen nous reptes que cal superar.

La present tesi es centra en la privadesa dels ciutadans com a usuaris de mitjans de transport vehiculars dins del marc d'una *e-society*. En concret, les contribucions de la tesi es focalitzen en les subcategories d'estacionament de vehicles privats en zones públiques regulades i en la realització de transbordaments entre línies intercomunicades en l'àmbit del transport públic.

Tradicionalment, els sistemes d'estacionament de vehicles en zones públiques, s'han basat en l'ús dels parquímetres. Després de pagar un import, en funció de la durada prevista de l'estacionament, es genera un tiquet difícilment associable amb el conductor. Actualment, aquest pagament es pot realitzar digitalment mitjançant una *app* instal·lada en un *smartphone*. Aquesta digitalització permet associar inequívocament un *e-ticket* amb el seu propietari i deduir-ne informació sensible.

Per altra banda, els sistemes de transport públic que permeten transbordaments entre línies interconnectades, es basen en la reutilització d'un bitllet de forma limitada durant un període preestablert. Existeixen múltiples propostes que permeten la compra i ús de *e-tickets* d'un sol ús que garanteixen la privadesa dels usuaris. Altrament, en les propostes que permeten la reutilització d'un *e-ticket* mitjançant transbordaments, la privacitat dels usuaris es veu compromesa després d'una inspecció *in-situ*.

Una anàlisi acurada de les dades recopilades pels proveïdors d'aquests serveis, sobre un determinat usuari, pot proporcionar informació personal sensible com per exemple: horari laboral, professió, *hobbies*, problemes de salut, tendències polítiques, inclinacions sexuals, etc. Tot i que existeixin lleis, com l'europea GDPR, que obliguin a utilitzar les dades recollides de forma correcta per part dels proveïdors de serveis, ja sigui a causa d'un atac informàtic o per una filtració interna, aquestes dades poden ser utilitzades per finalitats il·legals. Per tant, el disseny protocols que garanteixin la privadesa dels ciutadans que formen part d'una *e-society* esdevé una tasca de gran importància.

Resumen

El uso de dispositivos inteligentes, como los *smartphones*, interconectados a través de Internet es una realidad desde hace años. En una *e-society*, parte del entorno tradicional se ha visto reemplazado por el entorno digital. Este cambio, trae mejoras y beneficios sobre las sociedades de las que formamos parte pero también aparecen nuevos retos a superar.

La presente tesis se centra en la privacidad de los ciudadanos en el transporte vehicular dentro del marco de una *e-society*. En concreto, las contribuciones de la tesis se centran en las subcategorías de estacionamiento de vehículos privados en zonas públicas reguladas y en la realización de transbordos entre líneas interconectadas en el ámbito del transporte público.

Tradicionalmente, los sistemas de estacionamiento de vehículos en zonas públicas, se han basado en los parquímetros. Después de pagar un importe, en función de la duración prevista de estacionamiento, se genera un tiquet difícilmente asociable con el conductor. Actualmente, el pago se puede realizar digitalmente mediante una *app* instalada en un *smartphone*. La digitalización de este proceso suele asociar inequívocamente un *e-ticket* con su propietario y con toda la información deducible.

Por otra parte, los servicios de transporte público con transbordos entre líneas interconectadas, se basan en la reutilización de un billete de forma limitada durante un periodo preestablecido. Existen muchas propuestas que permiten la compra y uso único de *e-tickets* que garantizan la privacidad de los usuarios. De lo contrario, la privacidad de los usuarios se ve comprometida tras una inspección *in situ*, para las propuestas que permiten *e-tickets* reutilizables mediante transbordos.

Una análisi acurada de los datos recopilados por los proveedores de los servicios, sobre un determinado usuario, puede proporcionar información personal sensible como por ejemplo: horario laboral, profesión, *hobbies*, problemas de salud, tendencias políticas, inclinaciones sexuales, etc. A pesar que hay leyes, como la europea GDPR, que obligan a usar de forma correcta los datos recopilados por parte de los proveedores de servicios, ya sea por un ataque informático o por una filtración interna, estos datos pueden utilizarse para fines ilegales. Por lo tanto, es vital diseñar protocolos que garanticen la privacidad de los ciudadanos que forman parte de una *e-society*.

Statement

This thesis fulfills the requirements of the Universitat de Lleida to obtain the degree of Doctor in *Engineering and Information Technology*.

Acknowledgements

Prof. Francesc Sebé for guidance and patience.

Txell for love.

Family for letting me do what I want.

Teresa Salvia, *el Señorito* and *l'àvia Conxita* for the first steps.

Mr. Daniel Garnica for his vocation.

Prof. Josep Domingo for wanting a country beyond the *xupa-xups*.

COVID-19 for days of confinement.

Funding

This research was partly funded by the Spanish Ministry of Science, Innovation and Universities (project number MTM2017-83271-R).

This thesis has received a grant for its linguistic revision from the Language Institute of the Universitat de Lleida (2020 call).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter is a brief introduction to the general concepts, the main objectives, and the contributions of this thesis.

It first presents an introduction to smart cities. Next, the chapter focuses on vehicular systems and the privacy concerns that arise in such systems. After that, the objectives and contributions of the thesis are explained. Finally, the structure of the document is detailed.

1.1 Smart cities

The urban population of the world has grown from 751 million in 1950 to 4.2 billion in 2018. In 2018, 55.3% of the world population lived in urban areas, a proportion that is expected to increase to 60% by 2030. In 2018, there were 33 megacities (cities with 10 million inhabitants or more) and this figure is projected to rise to 43 in 2030 [Nat20a].

This situation has made it urgent to find smarter ways to manage large cities [CDN09]. The tendency of making cities smart makes them susceptible to including new methods of computerization which could be integrated into services with an already established infrastructure [KM16].

According to Caragliu *et al.* [CDN09] a city is smart when "*investments in human and social capital and traditional (transport) and modern (ICT) communication infrastructure fuel sustainable economic growth and a high quality of life, with a wise management of natural resources, through participatory governance*".

Some of the benefits of a city being smart are: efficient resource utilization, better quality of life, higher levels of transparency and openness [Nua+15]. The Centre of Regional Science (SRF) of Vienna University of Technology, in its research reported in [Gif+07], found that there are many fields of activity related to the smart city term. A smart city performs well the six key characteristics of urban development: smart economy, smart mobility, smart governance, smart living, smart people, and smart environment.

Table 1.1 illustrates the six characteristics and their assigned factors. Compliance with these characteristics indicates the degree of conversion to a smart city.

Several software applications have been developed within the scope of smart cities. For

Table 1.1: Characteristics and factors of a smart city

Characteristic	**Factors**
Smart Economy (Competitiveness)	Innovative spirit
	Entrepreneurship
	Economic image & trademarks
	Productivity
	International embeddedness
	Ability to transform
Smart Mobility (Transport and ICT)	Local accessibility
	International accessibility
	Availability of ICT-infrastructure
	Sustainable, innovative and safe transport systems
Smart Governance (Participation)	Participation in decision-making
	Public and social services
	Transparent governance
	Political strategies & perspectives
Smart Living (Quality of life)	Cultural facilities
	Health conditions
	Individual safety
	Housing quality
	Education facilities
	Touristic attractiveness
	Social cohesion
Smart People (Social and human capital)	Level of qualification
	Affinity to life long learning
	Social and ethnic plurality
	Flexibility
	Creativity
	Participation in public life
Smart Environment (Natural resources)	Attractiveness of natural conditions
	Pollution
	Environmental protection
	Sustainable resource management

example, smart buildings use applications to optimize the use of locally produced sustainable energy and the local grid capacity [Dhu+15; Mon+13]. Smart adaptive light applications can improve traffic flow and reduce CO_2 emissions and air pollution [Bod+14]. Other smart applications can monitor water pressure and detect water leaks in real time [BM15]. These software applications usually collect large amounts of data coming from different sources such as computers, smartphones, sensors, GPS, cameras, etc. This huge amount of information is analyzed by means of the so-called *big data* [KB15;

Nua+15] technology.

The volume of data generated by smart cities has grown exponentially. These data provide information, in many cases in real time, about the city and its citizens. The analysis of these data offers several advantages [Kit13]:

- It helps citizens to make day-to-day decisions.
- It helps governments to offer a more efficient and effective city.
- It helps corporations to generate new business opportunities.

On the contrary, the collection and correlation of this large amount of data allow the creation of detailed profiles of citizens. The collection of sensitive citizen data is a business case [Sch05] and is a problem in smartphone applications [Li+15]. This is the reason why smart cities are a major threat to the privacy of citizens.

Unfortunately, privacy protection is not usually within the indicators used to determine the *smartness* of cities. In [PR20], the term *privacy* only appears once. Fortunately, the use of technologies in a smart city is determined by their acceptance by the citizens. Therefore, the protection of citizens' privacy will be a condition imposed by users. *Privacy-friendliness* plays a key role in the success of the smart cities of the future [EW18].

1.2 Privacy in vehicular systems

Smart mobility is one of the characteristics of a *smart city*. It is dedicated to improving the efficiency of public transport and the management of private vehicles [PBS13]. Vehicular ad-hoc networks (VANETs) are a technology which allows traffic safety and efficiency to be increased, and other applications to be enabled in the domain of vehicular communication [Sch+08]. Many applications can be included under the terms of business and entertainment, like music downloading, fleet management, simpler vehicle maintenance or payment for parking or road usage (Table 1.2).

All this digitization opens up the possibility of massive data collection by automatic monitoring. Individual information about transport movements allows private information to be inferred, such as the place of residence and work, work schedule, hobbies, or even health problems.

On the other hand, privacy is a fundamental right explicitly stated in the Universal Declaration of Human Rights (1948), reflected in Article 12: *"No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks"* [Nat20b]. Legislation like the European GDPR [Com18] protects the citizens with regard to the processing of personal data collected by any organization. Although service providers are forced to comply with the law, such regulations cannot prevent misuses conducted by criminals that have broken into system servers.

For this reason, privacy must be a key aspect in the design of secure systems and protocols involving the mobility of people and vehicles. That is the case for electronic transport

Table 1.2: Overview on applications for VANETs

Category	**Situation/Purpose**	**Application examples**
Active safety	Dangerous road features	1. Curve speed warning 2. Low bridge warning 3. Warning about violated traffic lights or stop signals
	Abnormal traffic and road conditions	1. Vehicle-based road condition warning 2. Infrastructure-based road condition warning 3. Visibility enhancer 4. Work zone warning
	Danger of collision	1. Blind spot warning 2. Lane change warning 3. Intersection collision warning 4. Forward/rear collision warning 5. Emergency electronic brake lights 6. Rail collision warning 7. Warning about pedestrians crossing
	Crash imminent	1. Pre-crash sensing
	Incident occurred	1. Post-crash warning 2. Breakdown warning 3. SOS service
Public service	Emergency response	1. Approaching emergency vehicle warning 2. Emergency vehicle signal preemption 3. Emergency vehicle at scene warning
	Support for authorities	1. Electronic license plate 2. Electronic drivers license 3. Vehicle safety inspection 4. Stolen vehicle tracking
Improved driving	Enhanced driving	1. Highway merge assistant 2. Left turn assistant 3. Cooperative adaptive cruise control 4. Cooperative glare reduction 5. In-vehicle signage 6. Adaptive drivetrain management
	Traffic Efficiency	1. Notification of crash or road surface conditions to a traffic operation center 2. Intelligent traffic flow control 3. Enhanced route guidance and navigation 4. Map download/update 5. Parking spot locator service
Business/ Entertainment	Vehicle maintenance	1. Wireless diagnostics 2. Software update/flashing 3. Safety recall notice 4. Just-in-time repair notification
	Mobile services	1. Internet service provisioning 2. Instant messaging 3. Point-of-interest notification
	Enterprise solutions	1. Fleet management 2. Rental car processing 3. Area access control 4. Hazardous material cargo tracking
	E-payment	1. Toll collection 2. Parking payment 3. Gas payment

tickets [Mut+12], pseudonym management in vehicular networks [Pet+15], management of parking space [Yan+11; Hua+18], or vehicle location proof systems [Zha+15].

This thesis focuses on privacy in the following areas of vehicular systems:

- **Parking systems.** Parking systems can be classified into several categories: parking spot search systems, reservation systems, payment systems, etc. This thesis focuses on pay-by-phone parking systems for regulated public parking areas. Just after parking, the driver simply has to log into the *app* and indicate the car license plate

number, the area of the city she has parked in, and the expected duration. Parking enforcement officers should be able to determine the payment status of parked cars so that the presence of an online server with access to all the data about parking transactions is required. These data are highly sensitive since they allow private information about drivers to be inferred. So as to preserve the privacy of drivers, the only information that should be managed by the system is that making it possible to check whether or not appropriate payment for a given parked car has been made.

- **Transport systems.** This thesis has addressed public transport systems that allow transfers between connecting lines. In the digital era, the replacement of paper-based tickets with electronic ones (*e-tickets*) is a reality. Users, with the required *app* installed in their personal smartphones, can purchase an *e-ticket* to perform the desired trip. The information managed by the ticket issuer must not make it possible to trace the way a specific user uses public transport. Regarding data privacy, the *app* must guarantee anonymity and unlinkability among all the trips performed with a given *e-ticket*, even after an inspection.

1.3 Objectives

This thesis aims to contribute to the design of phone-based protocols that preserve the privacy of users in vehicular systems. We have analyzed current systems, designed new ones, and added new features to existing proposals.

More specifically:

- **Parking systems.** Existing privacy preserving pay-by-phone parking systems have been studied and analyzed in depth. Our objective in this area has been to design systems in which a unique payment is performed at the beginning of the parking operation. In this way, the mobile device of a driver may be out of coverage while her car is parked without causing any risk of being fined.
- **Transport systems.** We have performed an in-depth study and analysis of several existing phone-based e-ticket proposals that preserve the users' privacy. Our research in this area has focused on proposals that allow the reuse of an e-ticket. The main objective was to design an e-ticket system in which, even after an inspection, the transport system is only able to determine whether the user is allowed to perform the current trip.

1.4 Contributions

This thesis is composed of three contributions. As a result of them, three scientific papers were written. Two of them have been published in international scientific journals. The third was presented in an international conference on e-society privacy and published in the corresponding proceedings.

The abstract of the resulting papers together with a short description of their contributions are included below:

1. **"Parking tickets for privacy-preserving pay-by-phone parking"** published in Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society in London, United Kingdom [BS19].

   Abstract. *Traditionally, the payment required for parking in regulated areas has been made through parking meters. In the last years, several applications which allow these payments to be performed using a mobile device have appeared.*
   In this paper we propose a privacy-preserving pay-by-phone parking system offering the same privacy as the traditional paper-based method even assuming an internal attacker with full access to all the information managed by the system servers. Drivers' privacy is preserved without requiring them to trust any party. Furthermore, the system can tolerate that the mobile devices of drivers fall out of coverage while their cars are parked.

   Contribution. The paper presents a privacy-preserving pay-by-phone parking system in which the driver performs a one-time payment at the beginning of a parking operation. This avoids the need to perform periodic micro-payments (*e*.g. each 5 minutes) and therefore the mobile phone may become unavailable while the car is parked. In addition, drivers can recover the money corresponding to unused time if the parking operation takes less time than initially expected.

2. **"A Construction for Providing Reusability to Mobile Phone-Based e-Tickets"** published in the international open access journal *IEEE Access* [BS20a].

   Abstract. *Nowadays, the use of electronic tickets in public transport is a reality. Several mobile phone-based e-ticket systems have been proposed so far. Even so, very few of them include reusability in the sense that a single e-ticket allows several journeys to be made. Although the identity of users is usually hidden behind a pseudonym, the existing proposals providing reusability allow the system to link all the journeys made with a given e-ticket.*
   In this paper we present a privacy-preserving construction allowing a mobile phone-based e-ticket system to be endowed with reusability. The privacy of users is proven to be preserved even assuming an internal attacker with full access to all the information managed by the system servers. All the sensitive interactions of a user with the system remain anonymous and unlinkable. Further, as a result of an inspection, the system is only able to determine whether the inspected user is allowed to make the current journey.

   Contribution. We present an extension, or *plug-in*, that can be coupled to any existing privacy-enabled ticketing system so as to provide the *reusability* property. In this way, a limited number of trips can be made before an expiration time which is set at the moment of ticket validation. Regarding data privacy, the construction guarantees anonymity and unlinkability among all the trips made with a given e-ticket, even after an inspection. This aspect constitutes the main contribution of the paper.

3. **"An efficient privacy-preserving pay-by-phone system for regulated parking areas"** published in the international journal *International Journal of Information Security* [BS20b].

   Abstract. *Traditional pay-and-display ticket machines currently coexist, but will probably be replaced in the near future with pay-by-phone applications. Such applications facilitate the payment for parking in regulated areas. Companies providing this service collect and manage information about all the parking transactions performed by drivers. That information is very sensitive and can be used to generate reports on the parking history of drivers, posing a threat to their privacy.*

   This paper proposes a pay-by-phone parking system in which the service provider is prevented from being able to track the parking transactions of drivers. The new proposal requires drivers to be connected only at the beginning of a parking transaction, or at the moment of indicating that a parking transaction took less time than expected. Prototype experiments have shown that the new proposal is much more efficient, in terms of computational cost, than the most complete previously existing system, while providing the same functionalities and higher security.

   Contribution. The contribution consists of a privacy-preserving pay-by-phone parking system providing all the functionalities of [BS19] but avoiding the use of *cut-and-choose* checks. In this way, its computational cost is significantly lower. For instance, on the server side, the new proposal is approximately 8.5 times faster at generating e-coins and 80 times faster at issuing tickets, when compared with [BS19].

1.5 Structure of this thesis

This thesis has been divided into several chapters. Their content is summarized below:

- **Chapter 1.** This is an introduction to the scope of our research. It summarizes the objectives and contributions of the thesis.
- **Chapter 2.** This chapter introduces and explains the cryptographic primitives and tools used in the design of our proposals. The chapter also defines the notation used throughout this dissertation.
- **Chapter 3.** This is dedicated to the contribution presented in the research paper "*Parking tickets for privacy-preserving pay-by-phone parking*" [BS19]. The chapter begins with a description of previous related work. Next, it presents the system model designed and describes the proposed system together with a performance analysis.
- **Chapter 4.** This chapter presents the research in the paper "*A Construction for Providing Reusability to Mobile Phone-Based e-Tickets*" [BS20a]. First, the reader is introduced to the research area of privacy-preserving e-tickets for transport systems. Then, the system model and the novel proposed system are presented. Next, the privacy and security of our proposal are analyzed. The chapter is concluded with a performance analysis resulting from an implemented simulator.

- **Chapter 5.** This includes the contributions summarized in the article "*An efficient privacy-preserving pay-by-phone system for regulated parking areas*" [BS20b]. The chapter first details the aspects of [BS19] which needed to be improved. Next, the new proposal is described. Its security and performance are compared against [BS19].

- **Chapter 6.** The last chapter includes a summary of the thesis achievements. Some open issues for future research are suggested.

Chapter 2

Preliminaries

This chapter summarizes the cryptographic primitives and tools used in the design of the contributions of this thesis. It also introduces the formal notation used in this dissertation.

2.1 Hash functions

A hash function [NY89] is a cryptographic one-way function that, given an arbitrary message, M, provided as input, computes and returns a fixed-length (recommended to be between 160 and 512 bits long) digest of M as output. Given a hash function $\mathcal{H}$ and data M, we denote the resulting digest as $\mathcal{H}(M)$.

A cryptographic hash function must provide the following characteristics to be considered secure [Gol01]:

1. **Preimage resistance.** Given m, it must be computationally hard to find M such that $m = \mathcal{H}(M)$. A hash function for which preimages cannot be found efficiently is said to be *one-way* or *preimage resistant*.
2. **Second preimage resistance.** Given M, it must be computationally hard to find $M' \neq M$ such that $\mathcal{H}(M) = \mathcal{H}(M')$. A hash function for which second preimages cannot be computed efficiently is said to be *second preimage resistant*.
3. **Collision resistance.** It must be computationally hard to find two different messages, M and M', such that $\mathcal{H}(M) = \mathcal{H}(M')$. A hash function for which collisions are hard to find is said to be *collision resistant*.

The main cryptographic hash functions are summarized in Table 2.1.

In 1976, Diffie and Hellman identified the need for one-way hash functions [DH76]. The first definitions, analysis and constructions for cryptographic hash functions were given in the late 1970's. Rabin [Rab78] proposed a design with a 64-bit result based on the DES block cipher [NBS77]. The limitations of block cipher based hash functions resulted in a series of designs from scratch. During the 1980's the need for fast and secure hash functions became clear.

In 1989, Rivest developed an 8-bit Message-Digest Algorithm named MD2 [Kal92]. Then, in the early 1990's, Rivest proposed the MD4 [Riv91] and MD5 [RD92] functions.

Table 2.1: Summary of cryptographic hash functions

Algorithm (Variant)	**Year**	**Output**	**Security attacks**
MD4 [Riv91]	1991	128 bits	[Ste06; Wan+04b]
MD5 [RD92]	1992	128 bits	[Ste06; Wan+04b; Klí05]
HAVAL [ZPS93]	1993	128-256 bits	[Wan+04b]
SHA-0 [NIS93]	1993	160 bits	[WYY05b]
SHA-1 [NIS95]	1995	160 bits	[WYY05a]
SHA-2 (SHA-256) [NIS02]	2002	256 bits	[SS08]
SHA-2 (SHA-384) [NIS02]	2002	384 bits	[SS08]
SHA-2 (SHA-512) [NIS02]	2002	512 bits	[SS08]
SHA-2 (SHA-224) [NIS08]	2004	224 bits	[Aok+09]
SHA-2 (SHA-512/224) [NIS12]	2012	224 bits	[DEM15]
SHA-2 (SHA-512/256) [NIS12]	2012	256 bits	[DEM15]
SHA-3 (SHA3-224) [NIS15]	2015	224 bits	[DDS14a; Guo+19]
SHA-3 (SHA3-256) [NIS15]	2015	256 bits	[DDS14a; Guo+19]
SHA-3 (SHA3-384) [NIS15]	2015	384 bits	[DDS14a; Guo+19]
SHA-3 (SHA3-512) [NIS15]	2015	512 bits	[DDS14a; Guo+19]

MD5 has been one of the most widely used hash functions, although it is currently out of date due to serious security issues [Ste06]. In 2005, Vlastimil Klíma published an algorithm that can find an MD5 collision in 8 hours using a personal computer [Klí05].

In 1993, the NIST (National Institute for Standards and Technology, USA) proposed an improved version of MD5, called SHA (Secure Hash Algorithm), with a 160-bit result. This version is referred to as SHA-0 [NIS93].

In 1995, NIST discovered a weakness in SHA-0 [WYY05b], which resulted in a new release of the standard published under the name SHA-1 [NIS95].

In 2002, NIST published another family of hash functions commonly called SHA-2: SHA-256, SHA-384 and SHA-512 [NIS02], motivated by the discovery of new attacks on SHA-1 [WYY05a].

In 2005, NIST specified a new family of Secure Hash Algorithms named SHA-3 to substitute the SHA-2 family (although attacks against it were not published until a few years later [SS08; Aok+09; DEM15]). The SHA-3 family consists of four cryptographic hash functions: SHA3-224, SHA3-256, SHA3-384 and SHA3-512 [NIS15].

In 2007, NIST announced the SHA-3 Cryptographic Hash Algorithm Competition [20b]. In 2012, KECCAK [Ber+13] was the winning algorithm for the SHA-3 family. Since the KECCAK hash function was made public in 2008, there has been intensive crypto-analysis from the research community [DDS12; DDS14b; Din+15; GLS16].

Although all hash functions have been attacked in some way, some of the attacks cannot be considered to be practical at the moment. According to [Inf20], the following hash functions are considered to be cryptographically strong: SHA-256, SHA-512/256, SHA-384, SHA-512, SHA3-256, SHA3-384 and SHA3-512.

Hash functions are a powerful tool used for a variety of purposes [SG12]:

1. **Integrity and authentication.** Hash functions allow message authentication and integrity to be achieved without the use of symmetric encryption [BCK96a].
2. **Digital signatures.** Hash functions are employed in the scope of digital signature schemes [Gau07] to reduce the size of the data to be signed. The signer only signs the digest of the message.
3. **Authentication of users in computer systems.** Hash functions are used to store the digest of users' passwords in authentication systems. In this way, passwords remain secure even after an attacker obtains access to the user registry [LHH02].
4. **Digital timestamping.** Hash functions and digital signatures are used to implement digital time stamping [HS91].
5. **Pseudo random number generation.** Hash functions can be used as a building block in the implementation of Pseudo Random Number Generators [BCK96b; HHR06].
6. **Unique data identification.** Hash digests allow unique data to be identified, and duplicate information and data corruption to be detected [Yan+13].

2.2 HMAC functions

An HMAC function is a keyed cryptographic one-way function [BCK96a]. Given a message M and a key K, we denote the resulting digest as $\text{HMAC}_K(M)$. The relation between a message M and its HMAC digest can only be determined if key K is known.

Given a hash function $\mathcal{H}$, the algorithm for computing $\text{HMAC}_K(M)$, described in [KB00], is as follows:

- `B` $\equiv$ Byte-length of the blocks of data of the hash function.
- `L` $\equiv$ Byte-length of the hash output.
- `ipad` $\equiv$ Byte `0x36` repeated `B` times.
- `opad` $\equiv$ Byte `0x5c` repeated `B` times.
- K' is a fixed `B` bytes key assigned as:
  - If $|K| <$ `B` then $K' = K$ and append the byte `0x00` to K' until $|K'| =$ `B`;
  - Else if, $|K| >$ `B` then $K' = \mathcal{H}(K)$;
  - Else, $K' = K$.
- Compute

$$\text{HMAC}_K(M) = \mathcal{H}(K' \oplus \texttt{opad} || \mathcal{H}((K' \oplus \texttt{ipad}) || M)).$$

The security of an HMAC function lies in the security of the underlying cryptographic hash function, $\mathcal{H}$, and the size and randomness of key K. Also, the choice of the `opad` and `ipad` constants is important [PSW12].

According to present knowledge, it is recommended to take $\mathcal{H}$ as a hash function of the SHA-2 or SHA-3 families [Inf20].

2.3 Public key encryption schemes

Encryption schemes can be classified as:

1. **Symmetric encryption schemes.** The data to be protected are encrypted and decrypted using the same key. The main disadvantage of these schemes is the need to exchange the secret key between the sender and the receiver of data.
2. **Asymmetric encryption schemes.** The receiver of data has to be in possession of two keys, a public key and a private key. Then, data are encrypted under the public key, whereas decryption requires the private one. The main advantage of this approach is that the public key can be known by all parties, even by an attacker, so that no secret key needs to be exchanged secretly.

Figure 2.1 depicts the two types of encryption. Twofish, AES, Blowfish, DES, 3DES, Safer and IDEA are popular symmetric encryption schemes. On the other hand, RSA and ElGamal are widely known public key encryption schemes.

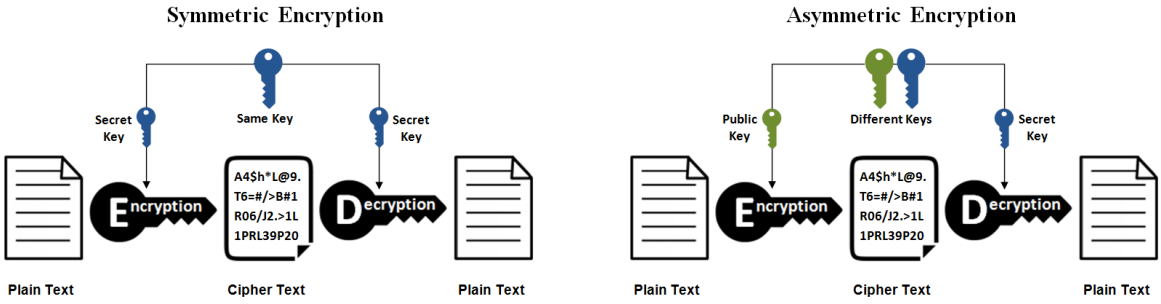

Figure 2.1: Symmetric/Asymmetric schemes
[Source: www.ssl2buy.com]

2.3.1 RSA encryption scheme

In 1976, Diffie and Hellman presented the idea of a public key (or asymmetric) encryption. The *Diffie–Hellman key exchange* protocol is a method for securely exchanging cryptographic keys over an insecure channel [DH76].

The protocol was followed shortly afterwards by L. R. Rivest, A. Shamir and L. Adleman who proposed a public-key cryptosystem called RSA [RSA78] in 1978. RSA is a widely known cryptosystem whose security holds on the assumed intractability of the Integer Factorization Problem (IFP) [Sch82].

If Alice has to send an encrypted message M to Bob, they must proceed as follows:

1. **Key generation.** Bob has to generate a private/public key-pair. An RSA public key is a tuple (N, e) composed of a hard-to-factor modulus $N = pq$ (p and q are large prime numbers) and a public exponent e, where $1 \leq e \leq \phi(N)$ and $\gcd(e, \phi(N)) = 1$. The RSA private key is an integer d that satisfies $de = 1 \pmod{\phi(N)}$. Bob sends his public key to Alice through any channel (whether or not secure).

2. **Encryption.** To encrypt the message M, with $M \in \mathbb{Z}_N^*$, Alice must use Bob's public key, (N, e). She computes $M' = M^e \pmod N$ and sends M' to Bob.
3. **Decryption.** Ciphertext M' is decrypted by Bob using his private key, d. He computes $M = M'^d \pmod N$ to obtain the original message M.

The security of the RSA encryption scheme lies on the unfeasibility of factoring the modulus N into its prime factors. According to [Inf20], N should be at least 2000 bits long ($\approx$ 2048 bits).

2.3.2 Optimal asymmetric encryption padding

Optimal asymmetric encryption padding (OAEP) is a procedure used to pad a short m-bit sequence to a longer n-bit one [BR95]. It is commonly used with RSA encryption to pad a message to RSA modulus length [Fuj+04]. The algorithm is schematically represented in Figure 2.2.

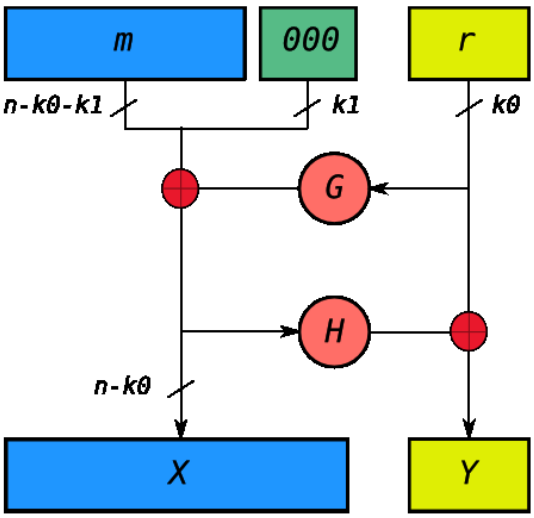

Figure 2.2: Optimal asymmetric encryption padding scheme
[Source: Wikipedia]

It uses a pair of cryptographic hash functions $\mathcal{G}$ and $\mathcal{H}$ to pad a message M so that its length becomes that of the RSA modulus. The scheme has two parameterized integers k_0 and k_1 so that $n = k_0 + k_1 + m$.

The $M' = \text{OAEP}(M)$ function returns a padded sequence M' from input M. The steps to encode M are described below:

1. Pad M with k_1 zeroes.
2. Generate a random r of k_0 bit length.
3. Seed $\mathcal{G}$ with r and generate $m + k_1$ pseudo-random bits, denoted by $\mathcal{G}(r)$.
4. Let $X = (M||0 \ldots 0) \oplus \mathcal{G}(r)$.
5. Seed $\mathcal{H}$ with X and generate k_0 pseudo-random bits, denoted by $\mathcal{H}(X)$.

6. Let $Y = r \oplus \mathcal{H}(X)$.
7. Return $M' = (X||Y)$.

The inverse function, $M = \text{OAEP}^{-1}(M')$, returns the original M from M'. The steps to decode M' are described below:

1. Compute $r = Y \oplus \mathcal{H}(X)$.
2. Recover $(M||0 \dots 0)$ as $X \oplus \mathcal{G}(r)$.

The OAEP scheme is a Feistel-like construction which generates a pseudo-random output by calling two internal hash functions. Any input to OAEP^{-1} not being the output of a previous call to OAEP produces a pseudo-random result.

After decoding, the *plaintext-awareness* property, with probability $1 - 2^{-k_1}$, is given. If such property is not required, we can set $k_1 = 0$. It is recommended that the cryptographic hash functions used, $\mathcal{G}$ and $\mathcal{H}$, have a digest length larger than 256 bits [Inf20].

2.4 Digital signatures

In 1976, Diffie and Hellman first described the notion of a digital signature [DH76]. A digital signature is a cryptographic scheme which demonstrates the authenticity of a message M transferred through an insecure channel.

The properties provided by a digital signature scheme are:

- **Authentication.** The receiver must be convinced that the message was signed by the sender.
- **Integrity.** The message has not been modified after being signed.
- **Non-repudiation.** The signer cannot deny having signed the message.

Signature schemes are built over an asymmetric key system. The private key is used to sign while the public one is used to verify the signature. Signature schemes are usually used in conjunction with a cryptographic hash function [Gol01]. Digital signature schemes are composed of the following algorithms [DK00]:

1. **Key generation.** The signer, S, is required to create a private/public key pair.
2. **Signature generation.** $\text{Sign}_S(M)$ denotes the resulting signature on message M computed using S's private key. Signatures are usually computed over the hash digest of the message, $\text{Sign}_S(\mathcal{H}(M))$, so as to reduce its computational cost.
3. **Signature verification.** The signed message can be verified later employing signer's public key.

There are multiple digital signature schemes that are based on different intractable problems. The most important ones are listed in Table 2.2.

Table 2.2: Digital signatures schemes

Scheme name	**Author**	**Intractable problem**	**Year**	**Reference**
RSA	R. Rivest, A. Shamir and L. Adleman	Integer Factorization (IFP)	1978	[RSA78]
DSA	US NIST	Discrete Logarithm (DLP)	1991	[NIS13]
Schnorr	C. P. Schnorr	Discrete Logarithm (DLP)	1990	[Sch90]
Okamoto	T. Okamoto	Integer Factorization (IFP)	1985	[OS85]
Feige-Fiat-Shamir	U. Feige, A. Fiat and A. Shamir	Quadratic Residue (QRP)	1988	[FFS88]
ECDSA	US NIST	Elliptic Curve Discrete Logarithm (ECDLP)	2005	[05]

2.4.1 DSA digital signature

The DSA digital signature scheme is a variant of the ElGamal signature scheme [ElG85]. It was proposed by the NIST in August 1991 to be used in its Digital Signature Standard [NIS13]. It is based on the difficulty of computing the DLP [McK90]. It has been revised four times since its release. Its last revision was performed in 2013. The signature generation procedure is faster than in RSA while its verification algorithm is slower.

Its protocols are defined as:

1. **Key generation.** First, three parameters (p, q, g) have to be set. Take two large primes p and q such that $p-1$ is a multiple of q. Then, select a generator g of the order q cyclic subgroup of $\mathbb{Z}_p^*$.
   A secret key V is generated by taking a random value in the $[0, q-1]$ range.
   A public key P is computed as $P = g^V \pmod{p}$.
2. **Signature generation.** A digital signature for message M is computed as follows:
   (a) Choose a random k, where $1 \leq k \leq q$.
   (b) Compute $R = g^k \pmod{p} \pmod{q}$.
   (c) Compute $S = k^{-1}(\mathcal{H}(M) + rx) \pmod{q}$.
   (d) The signature is the tuple $\{R, S\}$.
3. **Signature verification.** To verify a signature $\{R, S\}$ the receiver must proceed as follows:
   (a) Compute $w \equiv S^{-1} \pmod{q}$.
   (b) Compute $u_1 = \mathcal{H}(M)w \pmod{q}$ and $u_2 = Rw \pmod{q}$.
   (c) Compute $v \equiv (g^{u_1} y^{u_2} \pmod{p}) \pmod{q}$.
   (d) Verify whether $v = R$.

According to [Inf20], for a usage period up to and including 2022, the bit-length of the prime p should be at least 2000 ($\approx$ 2048 bits).

2.4.2 RSA digital signature

The RSA digital signature is based on RSA cryptosystem [RSA78]. RSA signatures can be verified very fast if the public exponent e of the public key is small.

The protocols composing this scheme are described below:

1. **Key generation.** An RSA public key is a tuple (N, e) composed of a hard-to-factor modulus $N = pq$ (p and q are large prime numbers generated randomly) and a public exponent e, where $1 \leq e \leq \phi(N)$ and $\gcd(e, \phi(N)) = 1$. The RSA private key is an integer d that satisfies $de = 1 \pmod{\phi(N)}$.
2. **Signature generation.** To sign the message M, where $M \in \mathbb{Z}_N^*$, the signer computes $S = M^d \pmod{N}$. The $\{M, S\}$ pair is a data-signature tuple.
3. **Signature verification.** To verify a signed message $\{M, S\}$ the receiver must compute and verify $S^e \pmod{N} = M$.

The security of the RSA scheme depends on the unfeasibility of decomposing the modulus N into its prime factors. According to [Inf20], N should be at least 2000 bits long ($\approx$ 2048 bits).

RSA simulated signatures

RSA digital signatures under a given public key (N, e) can be *simulated* by any party by taking an integer $S \in \mathbb{Z}_N^*$ and computing $M = S^e \pmod{N}$. The resulting pair $\{M, S\}$ is a valid data-signature tuple.

2.5 Blind signatures

In 1983, Chaum introduced the concept of blind signatures [Cha83]. Blind signatures are derivatives of digital signatures and have some additional features. The purpose of a blind signature protocol is to prevent the signer, Bob, from observing the message he signs and the signature; hence, he is later unable to associate the signed message with Alice [MOV01]. Blind signatures are fundamental for applications that guarantee user anonymity, *e.g.* e-cash, e-voting, direct anonymous attestation, and anonymous credentials.

A blind signature scheme consists of three parts [Cha83; Sch20]:

1. **Key generation.** The signer, Bob, generates a private/public key pair.
2. **Blind signature generation.** This is an interactive protocol between Bob and Alice:
   (a) Alice blinds the message M, with a random parameter r as $M' = \text{Blind}_r(M)$. M' is sent to Bob.
   (b) Bob signs the blinded message $\sigma' = \text{Sign}_{Bob}(M')$ with his private key and sends σ' to Alice.
   (c) Alice unblinds σ' to obtain σ, the signature on M.

3. **Blind signature verification.** Anyone can verify the signature σ over the message M using Bob's public key.

A blind signature scheme must guarantee two properties [JLO97; PS96]:

1. **Blindness.** It is unfeasible to link any valid message-signature pair (M, σ) to the instance of the blind signature generation protocol in which it was created.
2. **Unforgeability.** The only way to obtain a valid (M, σ) pair is executing the blind signature generation protocol with a signer holding a private key.

Several blind signature schemes have been designed over the past 35 years. These schemes are based on intractable problems such as Integer Factorization, Discrete Logarithm, Quadratic Residue, Bilinear Pairings, Lattices, etc. [Asg11].

In 1983, Chaum proposed a blind signature scheme based on RSA and the hardness of the IFP [Cha83].

In 1994, two blind signature schemes based on the DLP were proposed. The first one was based on the modified DSA [CPS94] while the second was based on the Nyberg-Rueppel signature scheme [NR93]. Regarding Nyberg-Rueppel-based schemes, they have suffered repeated attacks [LL00] and improvements since their first proposal. Several other DLP-based schemes have been proposed. In 1996, a blind signature protocol based on Schnorr signature scheme [Sch90] was proposed in [PS96]. In 2001, Schnorr improved the scheme by adding another computationally intractable problem, ROS, so that the scheme would remain secure. In 1998, an improved version of Okamoto-Schnorr blind signature scheme was proposed in [Poi98]. In 2002, Wagner described the first-known attack on Schnorr and Okamoto-Schnorr schemes [Wag02]. In 2019, Fuchsbauer *et al.* [FPS19], improved Schnorr's scheme to be secure against Wagner's attack [Wag02]. In 2000, a blind signature protocol based on ElGamal signature scheme [ElG85] was proposed in [MEE00] but it was proven to be weak in [HL01].

In 1998, Fan-Lei proposed a fast blind signature scheme based on the Quadratic Residue Problem [FL98]. In 2000, Shao proposed an improvement of Fan and Lei's scheme [Sha00], but a year later it was shown that Shao's scheme is vulnerable to Pollard-Schnorr attacks [FL01]. However, in 2002, Hwang, Lee and Lai [HLL02] showed that Fan-Lei's scheme did not provide the blindness property.

2.5.1 DSA blind signature

The proposal [CPS94], published in 1994, is the first blind signature scheme proposal based on the DLP. The scheme is designed as a variant of the DSA digital signature scheme (Section 2.4.1).

Its procedures are presented below:

1. **Key generation.** Bob generates a public/private key pair:
   (a) He chooses a large prime p so that $p-1$ has a large prime factor q, and takes g as primitive root of p.

(b) He chooses a random number x, where $x \in \mathbb{Z}_q$ and computes $y = g^x \pmod{p}$.
(c) Let y be the public key. Let x be the private key.

2. **Blind signature generation.** This is computed by means of the following interactive protocol:
   (a) Bob chooses a random $\tilde{k}$ where $\tilde{k} \in \mathbb{Z}_q$. Then, he computes $\tilde{R} = g^{\tilde{k}} \pmod{p}$. If $\gcd(\tilde{R}, q) = 1$ then he sends $\tilde{R}$ to Alice. Otherwise, Bob must choose another random $\tilde{k}$.
   (b) Alice receives $\tilde{R}$, checks $\gcd(\tilde{R}, q) = 1$, and randomly chooses $\alpha, \beta \in \mathbb{Z}_q$. Then, she computes $R = \tilde{R}^{\alpha} g^{\beta} \pmod{p}$. If $\gcd(R, q) = 1$ then she computes $\tilde{M} = \alpha M \tilde{R} R^{-1} \pmod{q}$ and sends $\tilde{M}$ to Bob; otherwise, Alice chooses another $\alpha, \beta \in \mathbb{Z}_q$.
   (c) Bob computes $\sigma' = \tilde{k}\tilde{M} + \tilde{R}x \pmod{q}$ and sends σ' to Alice.
   (d) Alice computes $\sigma = \sigma' R \tilde{R}^{-1} + \beta M \pmod{q}$ and $r = R \pmod{q}$.
   (e) The (r, σ) pair is a signature on M.
3. **Blind signature verification.** Anyone can verify a signature using Bob's public key y, by computing $T = (g^{\sigma} y^{-r})^{M^{-1}}$ and verifying $r = T \pmod{q}$.

A brief security overview of the DSA-based blind signature scheme is provided below:

1. **Blindness.** Intuitively, it is easy to see that two message-signature pairs $(M, (r, \sigma))$ and $(M', (\tilde{r}, \sigma'))$, where $\tilde{r} = \tilde{R} \pmod{q}$, are statistically independent of each other and cannot be linked to the random variables α and β.
2. **Unforgeability.** The generation of a valid message-signature pair $(M, (r, \sigma))$ is unfeasible because the scheme is based on the difficulty of solving the DLP.

As in the DSA digital signatures, the security of the DSA blind signature scheme depends on the length of p which should be at least least 2000 bits long ($\approx$ 2048 bits) [Inf20].

2.5.2 RSA blind signature

The first proposed blind signature scheme [Cha83] is based on RSA digital signatures (Section 2.4.2).

Its procedures are as follows:

1. **Key generation.** Bob generates a private/public key pair:
   (a) He chooses two random large primes p and q and computes $N = pq$ and $\phi(N) = (p-1)(q-1)$.
   (b) He chooses two integers e and d such that $ed \equiv 1 \pmod{\phi(N)}$ and $\gcd(e, \phi(N)) = 1$.
   (c) Let (e, N) be the public key.
   (d) Let d be the private key. He keeps the (p, q, d) tuple secret.
2. **Blind signature generation.** This consists of a three step protocol:
   (a) Alice chooses a random $r \in \mathbb{Z}_n$, verifies $\gcd(r, N) = 1$ and computes $M' = r^e \mathcal{H}(M) \pmod{N}$. She sends M' to Bob.

(b) Bob signs, under private key d, the blinded message as $\sigma' \equiv M'^d \pmod{N}$. He sends σ' to Alice.

(c) Alice unblinds the blinded signature σ' computing $\sigma \equiv \sigma' r^{-1} \pmod{N}$. The resulting σ is a digital signature over $\mathcal{H}(M)$.

3. **Blind signature verification.** Anyone can verify the signature σ over message M, using the Bob's public key (e, N), by verifying $\sigma^e = \mathcal{H}(M) \pmod{N}$.

A brief security overview of Chaum's RSA-based blind signature scheme is given below:

1. **Blindness.** The use of the random blinding factor, r, ensures that pair (M, σ) is statistically independent of pair (M', σ'), which can be viewed by Bob during the blind signature generation.
2. **Unforgeability.** If the hash function $\mathcal{H}$ is secure and the RSA problem is assumed to be unfeasible then this scheme is unforgeable. Complete proof can be found in [MSS98; Bel+08].

As with the RSA digital signature, the security of the RSA blind signature scheme requires a long length for N. According to [Inf20], N should be at least 2000 bits long ($\approx$ 2048 bits).

2.6 Partially blind signatures

A partially blind signature scheme is an extension of blind signature schemes that allows a signer, Bob, to explicitly include necessary information (expiration date, collateral conditions, or whatever) in the resulting signatures under some agreement with the message owner, Alice.

Generally, a partially blind signature scheme consists of three protocols [AO00]:

1. **Key generation.** A signer, Bob, generates a private/public key pair.
2. **Partially blind signature generation.** This is an interactive protocol between Bob and Alice. The public input is the agreed information $Info$. The private input by Alice is the hash of message M while the private input by Bob is his private key. At the end of the protocol Alice obtains the tuple $(Info, M, \sigma)$ where M and the signature σ are only known by her. We will denote $\sigma = \text{PartialSign}_{Bob}(M, Info)$.
3. **Partially blind signature validation.** Anyone can verify the signature σ over the message M and the agreed information, $Info$, using Bob's public key.

In 1996, the notion of partially blind signatures was introduced in [AF96]. It was not until the year 2000 that a partially blind signature scheme was published that gave proof of completeness, partial blindness and unforgeability assuming the hardness of the DLP [AO00].

In 2004, a partially blind signature scheme was proposed. It employed bilinear pairings that give signatures of short size [CYC04]. In the same year a new proposal was based on

the DLP and the Chinese Remainder [HC04]. A year later it was shown that this latter scheme was not secure [ZC05].

In 2005, the identity-based (ID-based) restrictive blind signature scheme from bilinear pairings was proposed [CZL05]. A few years later, the scheme was shown not to be secure [HH08].

In 2007, [LSK07] presented a scheme based on the Schnorr signature scheme, which is used for applications due to its reduced computation cost.

In 2012, Blazy *et al.* [BPV12] presented a protocol in which the public part is chosen by the signer avoiding the agreement between the parties before running the protocol.

In 2018, a new scheme based on chaotic map and factoring problems [TIA18] was proposed. It is much more efficient than the previous proposals like [TSI08].

2.6.1 WI-Schnorr partially blind signature

Abe and Okamoto proposed a partially-blind signature scheme in [AO00], based on the Schnorr signature scheme [Sch90]. Its security is DLP-based.

The procedures to sign a message M with the agreed information $Info$ are:

1. **Key generation.** Bob generates a private/public key pair.
   (a) He chooses two random large primes p and q such that $q|p-1$.
   (b) Let $g \in {\mathbb{Z}_p}^*$ such that g has order q. Let $\langle g \rangle$ denote a subgroup in $\mathbb{Z}_p^*$ generated by g.
   (c) He chooses public hash functions $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$ and $\mathcal{F} : \{0,1\}^* \to \langle g \rangle$.
   (d) Let $x \in \mathbb{Z}_q$ be the private key.
   (e) Let $y = g^x \pmod p$ be the public key.
2. **Partially blind signature generation.** This is an interactive four-step protocol:
   (a) Bob randomly chooses $u, s, d \in \mathbb{Z}_q$. Then,
      i. He generates the hash on $Info$ as $z = \mathcal{F}(Info)$.
      ii. He computes $a = g^u$ and $b = g^s z^d$.
      iii. Finally, he sends a and b to Alice.
   (b) Alice randomly chooses $t_1, t_2, t_3, t_4 \in \mathbb{Z}_q$. Then,
      i. She computes the hash digest of $Info$ as $z = \mathcal{F}(Info)$.
      ii. She computes $\alpha = a g^{t_1} y^{t_2}$ and $\beta = b g^{t_3} z^{t_4}$.
      iii. She computes the hash digest $\epsilon = \mathcal{H}(\alpha||\beta||z||M)$.
      iv. Finally, she computes $e = \epsilon - t_2 - t_4 \pmod q$ and sends e to Bob.
   (c) Bob computes $c = e - d \pmod q$ and $r = u - cx \pmod q$. Then, he sends the tuple $\{r, c, s, d\}$ to Alice.
   (d) Alice, computes
      i. $\rho = r + t_1 \pmod q$.
      ii. $\omega = c + t_2 \pmod q$.
      iii. $\sigma = s + t_3 \pmod q$.
      iv. $\delta = d + t_4 \pmod q$.

      Finally, she stores the signature as a tuple $\{\rho, \omega, \sigma, \delta\}$.

3. **Partially blind signature validation.** A signature is valid if it satisfies $\omega + \delta \equiv \mathcal{H}(g^{\rho}y^{\omega}||g^{\sigma}||\mathcal{F}(Info)^{\delta}||\mathcal{F}(Info)||M) \pmod{q}$.

2.7 Elliptic curves

An elliptic curve E over a prime finite field $\mathbb{F}_p$ is defined using the Weierstraß general equation as

$$Y^2 + A_1XY + A_3Y = X^3 + A_2X^2 + A_4X + A_6; \quad A_i \in \mathbb{F}_p.$$

The graphical representation of two elliptic curves defined over $\mathbb{R}$ can be seen in Figure 2.3. If the characteristic of $p \notin \{2, 3\}$ then the elliptic curve can be expressed with the reduced Weierstraß equation,

$$Y^2 = X^3 + AX + B,$$

with $A, B \in \mathbb{F}_p$ and $4A^3 + 27B^2 \neq 0 \pmod{p}$ [HM05; Tra06].

The set points of $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ satisfying expression E together with the point at infinity, $\mathcal{O}$, are denoted as $E(\mathbb{F}_p)$. An addition operation can be defined over $E(\mathbb{F}_p)$ which endows it with an abelian group structure with $\mathcal{O}$ being the identity element.

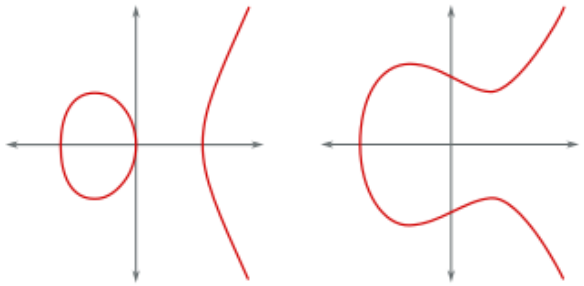

Figure 2.3: Elliptic curves $y^2 = x^3 - x$ and $y^2 = x^3 - x + 1$
[Source: Wikipedia]

It is easy to see that if $(x, y) \in E(\mathbb{F}_p)$ then $(x, -y)$ is also in $E(\mathbb{F}_p)$. Therefore, a point $(x, y) \in E(\mathbb{F}_p)$ can be expressed as (x, b), with b being a boolean that indicates whether $y > -y$. A compressed point (x, b) is expanded by computing the two square roots of $x^3 + Ax + B \pmod{p}$ and selecting one of them as indicated by bit b.

An addition operation can be defined over the set $E(\mathbb{F}_p)$ using the chord-tangent method. Given two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, where $P, Q \in E(\mathbb{F}_p)$ and $x_P \neq x_Q$, the result point, $R = (x_R, y_R)$, given by $R = P + Q$ is calculated as follows:

$$\begin{cases} x_R = \lambda^2 - x_P - x_Q \\ y_R = \lambda(x_P - x_R) - y_P \end{cases} \quad \text{where } \lambda = \frac{y_P - y_Q}{x_P - x_Q}.$$

Figure 2.4 represents the addition operation over two points of an elliptic curve.

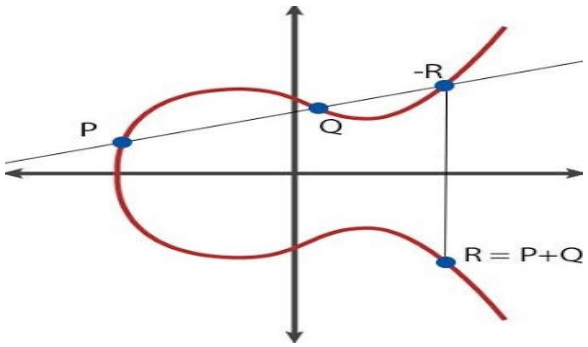

Figure 2.4: Elliptic curve point addition $R = P + Q$
[Source: www.medium.com]

The opposite point of $P = (x, y)$ is $-P = (x, -y)$. The scalar multiplication can be calculated as $Q = nP = P + \overset{n}{\cdots} + P$, where $n \in \mathbb{N}$. The Elliptic Curve Discrete Logarithm Problem (ECDLP) is a computationally hard problem [HM05]. Given two known points, $P, Q \in E(\mathbb{F}_p)$, the problem aims to find an integer $n \in \mathbb{N}$ which solves the equation $Q = nP$ [HM05].

2.7.1 Elliptic curve cryptography

Elliptic curve cryptography (ECC) schemes were proposed independently, in 1985, by Koblitz [Kob87] and Miller [Mil85]. Their security holds on the apparent intractability of ECDLP [HM05]. The use of ECC has become widespread due to the fact that it allows smaller key sizes to be employed. Table 2.3 compares the size ratios between ECC and RSA keys for equivalent security levels.

Table 2.3: Key size comparison between ECC and RSA

Bit level	**ECC**	**RSA**	**Ratio**
80	160	1024	1/6
112	224	2048	1/9
128	256	3072	1/12
192	384	7680	1/20
256	512	15360	1/30

The Elliptic Curve Integrated Encryption Scheme (ECIES) [GHS10] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [05] are elliptic curve-based encryption and digital signature schemes, respectively.

The ECIES encryption scheme is composed of the following procedures:

1. **Set-up.** Bob and Alice must choose:
   (a) An elliptic curve E whose set $E(\mathbb{F}_p)$ has a cardinality divisible by a large prime p. An order-p point P of $E(\mathbb{F}_p)$ is also chosen.
   (b) A Key Derivation Function (KDF) constructed from a hash function $\mathcal{H}$.

   (c) An encryption/decryption function for a symmetric-key encryption scheme (E/E^{-1}).
   (d) A Message Authentication Code (MAC) such as $HMAC$.
2. **Key generation.** A private key is set by taking a random integer k_B in the $[0, p-1]$ range. The corresponding public key is given by point $K_B = k_B P$.
3. **Encryption.** Alice encrypts a message M for Bob by using Bob's public key K_B following the steps below:
   (a) Randomly select $r \in [1, n-1]$ and compute $R = rG$.
   (b) Derive a shared secret $S = P_x$ where $P = (P_x, P_y) = rK_B$ and $P \neq \mathcal{O}$.
   (c) Derive keys as $k_E||k_M = KDF(S)$.
   (d) Encrypt the message $c = E_{k_E}(M)$.
   (e) Compute $d = HMAC_{k_M}(c)$ and send $\{R, c, d\}$ to Bob.
4. **Decryption.** Bob decrypts a ciphertext $\{R, c, d\}$ using his private key as follows:
   (a) Derive a shared secret $S = P_x$ where $P = k_B R$.
   (b) Derive keys $k_E||k_M = KDF(S)$.
   (c) Check $d = HMAC_{k_M}(c)$. If the previous checking is satisfied, he decrypts $M = E^{-1}_{k_E}(c)$.

The algorithm ensures an n bit security level for $p \geq 2^{2n}$, with p being the order of point P. According to [Inf20], n should be at least 240 bits long ($224 \lessapprox n \lessapprox 256$ bits).

2.7.2 Embedding elliptic curve points in $\mathbb{Z}_N$

Let E be an elliptic curve defined over $\mathbb{F}_p$, with p being a 224 bits prime, and let N be a 2048 bits RSA modulus. Although there are multiple procedures for embedding an elliptic curve point $P = (x, y) \in E(\mathbb{F}_p)$ into an integer $M' \in \mathbb{Z}_N$, such as [SSA17], below we detail a procedure designed by us.

Let us denote this procedure as Embed(P):

1. Represent $P = (x, y)$ in compressed form (x, b) and compose the 225 bits sequence $M = x||b$.
2. Set $m = 225$, $k_0 = 1823$, $k_1 = 0$ (our proposal does not require plaintext-awareness), and $n = 2048$, and run OAEP(M), to pad M into a 2048 bits long string M'.
3. If $M' \geq N$ then run OAEP(M) again taking a different random seed until $M' < N$.
4. Return M' as an element of $\mathbb{Z}_N$.

The reverse procedure, denoted Embed$^{-1}(M')$, returns the original elliptic curve point P from M' as:

1. Unpad M' into M by computing $M = \text{OAEP}^{-1}(M')$.
2. Compose the compressed point (x, b) by parsing M.
3. Uncompress (x, b) into the elliptic curve point (x, y).
4. Return $P = (x, y)$.

If M' was taken at random, the previous procedure may fail at step 3 if the elliptic curve E does not have any point with such an x-coordinate. The probability of such failure is about 1/2.

2.8 Range proofs of committed values

In 1988, Brassard, Chaum and Crepeau introduced the concept of commitment scheme in the context of the problem of flipping a fair coin over the phone [GC88]. A commitment scheme [Gol01] is a two-stage cryptographic primitive. In the commit stage, Alice commits a chosen value while keeping it hidden from Bob. In the opening stage, Alice reveals the committed value to Bob.

In general, Alice commits E to Bob as follows:

1. **Commit.** Alice chooses a random parameter c and commits E; so that $\text{Commit}_c(E)$ denotes the resulting commitment.
2. **Open.** To open the committed value, Alice reveals the parameter c to Bob.

Any commitment scheme must meet the following properties:

1. Alice cannot change the committed value.
2. Bob obtains no information about the committed value.
3. Alice can convince anyone about the committed value when she reveals it.

Some commitment schemes allow it to be proven in zero-knowledge that a committed number lies in an interval. The first approach to prove that a committed integer lies in a specific interval was published in [Bri+88].

The [Bou00] proposal allows it to be proven in zero-knowledge that a committed number lies in an interval $[a, b]$ efficiently. Given a $\text{Commit}_c(E)$, Alice can prove to Bob in zero-knowledge that $a \leq E \leq b$ without revealing E. The proposal is built on the Fujisaki-Okamoto commitment scheme [FO98].

2.9 Trusted timestamping

A trusted timestamping service is composed of a timestamp authority (TSA) which provides signed evidence about the existence of data M at a given *Datetime* [Ada+01].

To generate a timestamp over M [HS91], the TSA concatenates the current time to $\mathcal{H}(M)$ and calculates another hash of this concatenation as $\mathcal{H}(Datetime||\mathcal{H}(M))$. Finally, this hash is signed with the private key of the TSA as $\text{Sign}_{TSA}(\mathcal{H}(Datetime||\mathcal{H}(M)))$.

To verify the signature, the client must assure that TSA signs the hash message of M with the correct time included.

Chapter 3

Pay-by-Phone parking system

The content presented in this chapter was summarized in the scientific paper *"Parking tickets for privacy-preserving pay-by-phone parking"* presented in the *18th ACM Workshop* (2019) in London, United Kingdom, and published in the proceedings of the mentioned conference [BS19]. The proposed system has been designed to be implemented as a mobile application.

3.1 Introduction

In the late 1920s, Roger W. Babson filed several parking meter patents [Bab34]. In most proposals, after inserting coins into a pay station, a paper ticket to be placed on the dashboard of the car is issued.

The massive deployment of smart devices has facilitated the development of applications allowing such payments to be performed through the mobile phone [20c; 20d; 20e; 20f; 20j; 20h; 20a; 20i]. Upon parking, the driver simply has to log into the mobile app and indicate the car license plate number, the area of the city she has parked in, and the expected duration. The amount to pay is then deducted from a pre-paid balance, or charged directly on the driver's credit card.

Parking officers check the parking status of a car by typing its plate number into a mobile device which indicates whether or not a valid payment has been made. This requires the presence of an online server, accessible from parking officers' devices, with access to the data allowing the payment status of cars to be determined. These data are highly sensitive since they allow private information to be inferred about drivers, such as their work schedules, hobbies, or even health problems. To avoid unnecessary risks, the only information that should be managed by the system is that allowing a parking officer to check whether or not an appropriate payment for a given parked car has been made.

Privacy is a key aspect in the design of secure systems and protocols involving the mobility of people and vehicles. That is the case for electronic transport tickets [Mut+12], pseudonym management in vehicular networks [Pet+15], management of parking space [Yan+11; Hua+18], or vehicle location proof systems [Zha+15].

To avoid security breaches from inside the service provider, pay-by-phone parking sys-

tems should be designed by considering it as a party which might try to infer information about drivers' parking habits. This excludes checking the payment status of a car just from its plate number. In that case the service provider could simply query the system periodically with a targeted plate number and obtain accurate information about the time periods for which the traced car has been parked. Hence, the payment status of a car has to be determinable only from one-time pseudonyms which can only be obtained by a parking officer located close to the car. This makes it necessary to include some kind of on-board device providing one-time pseudonyms when requested by a parking officer [Pér15; GMS17].

3.2 Related work

Nowadays, there are several pay-by-phone parking systems [20c; 20d; 20e; 20f; 20j; 20h; 20a; 20i], but none of them addresses driver privacy. These mobile applications collect and store accurate data about each parking operation so that the generation of reports about parking habits can be performed in a straightforward manner.

These mobile applications are currently developed for the most important platforms: Android (Google) and IOS (Apple). Some of them can also be executed on obsolete BlackBerry devices [20d; 20f; 20a].

All these applications, except [20c], use a start-duration method in which the drivers are required to estimate the time during which the car will be parked. In [20c], a start-stop method is used in which the driver indicates the end of parking when removing the car.

Regarding the payment method, all the applications allow payment by credit card. The PayPal electronic payment method [20g] is supported by some of them [20c; 20a; 20i]. Finally, proposal [20d] uses its own payment method called *Parkmobile wallet*. In [20i] payments can also be made by entering a bank account number. Table 3.1 summarizes the features of several pay-by-phone parking system applications.

Table 3.1: Summary of pay-by-phone parking system applications

Parking system	**Payment**			**Platform**			**Parking**
	Credit card	**PayPal**	**Others**	**Android**	**IOS**	**BlackBerry**	**period**
ParkRight [20e]	✓	✗	✗	✓	✓	✗	start-duration
Pango [20c]	✓	✓	✗	✓	✓	✗	start-stop
Parkmobile [20d]	✓	✗	✓	✓	✓	✓	start-duration
PayByPhone [20f]	✓	✗	✗	✓	✓	✓	start-duration
PayStay [20h]	✓	✗	✗	✓	✓	✗	start-duration
Telpark [20j]	✓	✗	✗	✓	✓	✗	start-duration
Elparking [20a]	✓	✓	✗	✓	✓	✓	start-duration
RingGo [20i]	✓	✓	✓	✓	✓	✗	start-duration

Recent research works [Pér15; GMS17] have proposed systems that consider driver privacy. Both proposals share a similar system model. Pre-paid e-coins are used for

anonymous payments. In addition, an RFID-enabled device is to be placed in cars and queried by parking officers when checking the parking status of cars.

In [Pér15], when a driver parks her car, an anonymous e-coin payment for the expected parking duration is made. That payment is linked to a random identifier which is stored in the on-board device. When a parking officer checks a car, he queries its on-board device to obtain the random identifier which is then sent to the system server to determine whether a payment linked to it has been made. At that time, the parking officer can link the car license plate number to the current identifier, and from the data stored on the system servers, that identifier can be linked to the start and end times of the parking operation. Hence, the exact start and end times of the parking operation of that car are determined.

The proposal in [GMS17] provides better privacy: the parking officer, when checking the parking status of a car, only obtains a boolean indicating whether or not a valid payment for the checked car has been made. That proposal performs periodic micropayments for short-time intervals while the car is parked. A payment can only be linked to a plate number after querying the on-board device. In that case, the car can only be linked to the payment performed for the current short-time interval so that the start and end times are kept secret.

The system proposed in [GMS17] was the most complete privacy-preserving pay-by-phone parking system when our research began. Unfortunately, if the driver's mobile device could not perform some of the micropayments due to a lack of coverage, low battery or any other cause, the driver could be fined. Our proposal provides all the features of [GMS17] and, additionally, solves the mentioned drawback.

3.3 System and adversary models

In the proposed system, payments are performed for short-duration time intervals (*e.g.* 5 or 10 minutes). As a result of paying for a given time slot, the driver receives a ticket for that slot.

When a parking operation begins, the driver pays and obtains a ticket for each of the time slots composing the expected parking time. Tickets are paid using pre-paid e-coins. So as to preserve privacy, the e-coin system employed must be untraceable, like [Cha83]. This excludes the use of transferable cryptocurrencies with a publicly available transaction history [Nak09; But14; Pro20]. In our proposal we had to design an ad-hoc e-coin system able to deal with *valued* and *no-valued* e-coins. Finally, the unused tickets can later be revoked in advance.

3.3.1 System model

The system model (Figure 3.1) is composed of the following actors:

1. **Mobile application** (or **app**). This is run on the mobile device of drivers. It allows them to acquire pre-paid credit (in the form of e-coins), request tickets,

revoke unused tickets, and provide a ticket when requested by a parking officer.

2. **System server.** This is an on-line platform accessed by the mobile application to manage parking operations, and by parking officers to check the payment status of cars.
3. **On-board device (RFID).** This is placed inside cars and queried via RFID by a parking officer during an inspection.
4. **Parking officer.** He patrols regulated parking areas, queries the on-board device of cars, and asks the system server about their parking status.
5. **Timestamp authority** or **TSA.** This issues timestamps upon request by the system server.

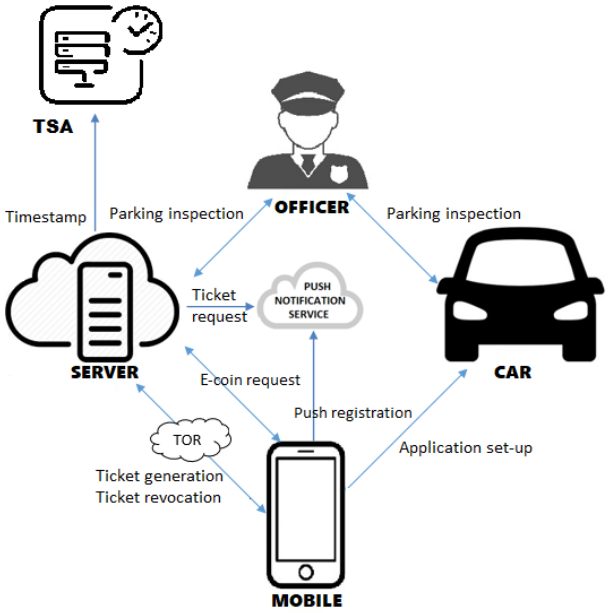

Figure 3.1: System model

A *push notification service* [BM16; RK13] allows data to be sent from servers to mobile applications. In our system, the mobile application requests an `ID_PUSH` identifier which is sent to the server together with the car plate number. In this way, the server will be able to send messages to the corresponding app. To avoid linking parking payments from IP addresses, an *anonymous channel* [19c] is used. It masks the IP address from which a client accesses a server.

3.3.2 Adversary model

The adversary model considered equals the one described in [GMS17]:

1. The mobile application and the on-board device cannot be corrupted.

2. System servers and parking officers will follow the protocol steps as specified but may collaborate with an adversary by providing all the data to which they have access.

From the service provider's point of view, drivers are untrusted parties which might try to obtain parking time without paying for it. Drivers may intentionally execute a modified protocol in order to obtain beneficial results. This attack could be carried out collaboratively by several drivers.

3.3.3 Design objectives

In this section we describe the design objectives. Privacy objectives are equivalent to those in [GMS17]. An additional objective regarding system tolerance to mobile phones going out of coverage has been added.

1. Tickets can only be linked to a car plate number as a result of a parking status check performed by a parking officer located close to the car.
2. The information provided by a driver during a parking inspection only allows it to be determined whether or not a ticket for the current time exists.
3. During a parking operation, the mobile app only needs an Internet connection at the beginning and at the moment of revoking unused tickets.

Objective 1 implies that the parking status of a car cannot be tracked automatically (a parking officer has to be in situ).

Objectives 1 and 2 determine that the information collected by the system provides an advantage to the creation of parking profiles. The information an attacker can obtain is exactly the same as it would obtain by roaming parking places and collecting information about parked cars.

Objective 3 was not addressed in [GMS17]. It states that a driver will not be fined due to her phone being out of coverage during a parking inspection.

3.4 System proposal

3.4.1 System overview

A driver installs an app into her mobile device which manages an electronic wallet that is loaded with pre-paid e-coins (which can be *valued* or *no-valued*). E-coins are acquired by running the "E-coin request" procedure. The price of a valued e-coin corresponds to the price for parking during a time slot. These e-coins are purchased in batch and paid via some online payment method like credit card. No-valued e-coins are free of charge.

Upon parking, a driver generates a ticket for each of the (consecutive) slots that compose the expected parking time by running the "Ticket generation" procedure. Each ticket is paid anonymously via a valued e-coin and stored in the mobile app.

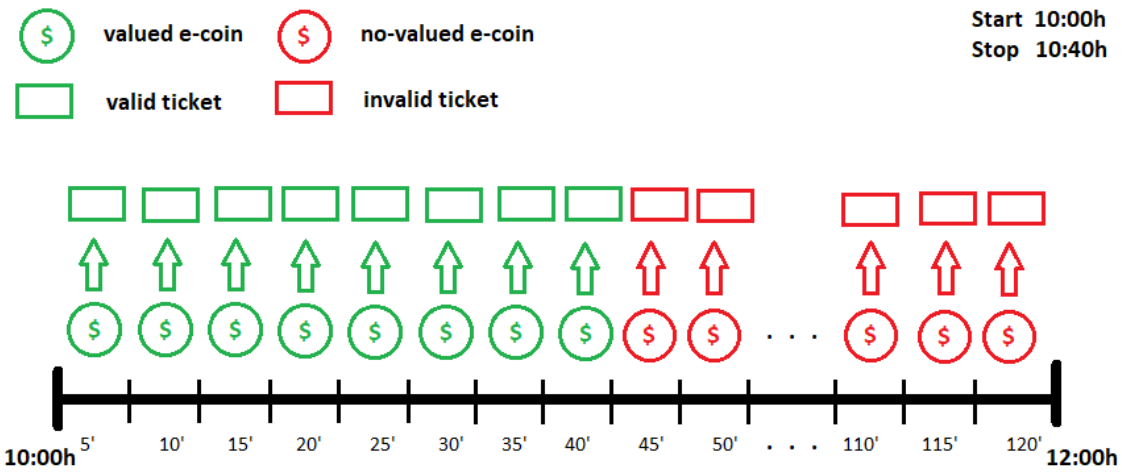

Figure 3.2: Abstract graphic

So as to hide the expected parking duration, the app always requests the same amount of tickets. Those tickets in which the driver expects to be parked are paid through valued e-coins while the remaining dummy ones are paid with no-valued ones (Figure 3.2).

When an inspection takes place, the parking officer, by running the "Parking inspection" procedure, first queries the on-board device of a car and then asks the system server to determine the parking status of that car. The system server then runs the "Ticket request" procedure so as to ask the mobile application for the corresponding ticket.

If the car is removed earlier than expected, unused valued tickets can be revoked through the "Ticket revocation" procedure. A valued e-coin is provided in exchange for each revoked ticket.

3.4.2 System description

This section details the procedures that compose the proposed privacy-preserving pay-by-phone parking system.

System parameters generation

This procedure must be run by the service provider before deploying the service:

1. Generate an RSA [RSA78] key pair for the system server. Let V_S be the private key (only known to the server) and let P_S be the public key.
2. Generate an RSA key pair for the timestamp server. Let V_{TSA} be the private key (only known to the timestamp server) and let P_{TSA} be the public key.
3. Generate a tuple (p,q,g) of DSA [NIS13] public set-up parameters which are stored at the system server.

Remark. The system server will use its key pair to (blindly) sign e-coins and tickets

during their issuance, thus preventing malicious drivers from being able to forge them. Timestamps will be issued to avoid disputes regarding the revocation status of tickets.

Application setup

A driver must install the app into her mobile device and enter the car license plate number and a credit card number (necessary to pay for the purchased e-coins). The app will then connect to the on-board device and send the plate number and the current time. The on-board device then sets its internal clock and generates an RSA key pair: V_D (private) and P_D (public). The public key P_D is transmitted to the mobile app.

Remark. The on-board device key pair will be used to generate signed messages which are returned to parking officers after being queried by them. Possession of such a signed message proves that a parking officer was located close to the car when a parking inspection took place.

E-coin request

The app includes a wallet which stores e-coins that are generated through a protocol run between the app and the system server. A *valued* e-coin has a price while *no-valued* ones are free of charge.

A *valued* e-coin is generated as follows:

1. The app performs a credit card payment for the requested e-coin[1].
2. The app generates a DSA key pair V_C/P_C over the (p, q, g) parameters.
3. The app generates an *even* number $x \in [0, (q-1)/2)]$.
4. The app computes $\mathcal{H}(P_C||g^x)$ and asks the server to compute an RSA blind signature on it.
5. The app consequently obtains $\text{Sign}_S(\mathcal{H}(P_C||g^x))$.

A *no-valued* e-coin is generated as follows:

1. The app generates $\mathtt{N}$ DSA key pairs $\{V_{Ci}/P_{Ci}\}_{0 \leq i < \mathtt{N}}$ and $\mathtt{N}$ *odd* numbers $\{x_i\}_{0 \leq i < \mathtt{N}}$ $\in [0, (q-1)/2)]$.
2. Then, the app computes $\{H_i = \mathcal{H}(P_{Ci}||g^{x_i})\}_{0 \leq i < \mathtt{N}}$ and blinds each H_i with a different blinding factor r_i. All the $\mathtt{N}$ blinded hash digests are sent to the server.
3. The server randomly chooses an index $j \in [0, \mathtt{N}-1]$ and sends it to the app.
4. For each $i \neq j$, the app sends r_i, x_i and P_{Ci} to the server.
5. The server unblinds the $\mathtt{N}-1$ hash digests H_i and checks whether each H_i equals $\mathcal{H}(P_{Ci}||g^{x_i})$. It also verifies that each x_i is odd and falls in the $[0, (q-1)/2)]$ range.
6. If all the checks are satisfied, the server blindly signs the remaining blinded hash and returns the result to the app.
7. The app consequently obtains $\text{Sign}_S(\mathcal{H}(P_{Cj}||g^{x_j}))$.

[1]E-coins will usually be generated in batch so that just a single credit card payment for the overall amount will be made.

Remark. The difference between valued and no-valued e-coins is the parity of x. Before issuing a no-valued e-coin, the system server must check that the corresponding x is odd through the described cut-and-choose technique. E-coins are blindly signed so that their issuance and spending cannot be related. Regardless of whether they are valued or no-valued, for each issued e-coin the app stores a tuple $\{\text{Sign}_S(\mathcal{H}(P_C||g^x)), V_C, x\}$.

Ticket generation

The driver has to issue a (valued) ticket for each of the time slots that compose the expected parking period. To hide its duration, the app always requests the same amount, R, of tickets (embracing the maximum allowed parking time). The tickets for slots in which the driver expects to be parked are paid with valued e-coins. The remaining ones are issued with no-valued ones so that no-valued dummy tickets are generated. This procedure is run through an anonymous channel:

1. The app takes a (valued or no-valued) e-coin $\{\text{Sign}_S(\mathcal{H}(P_C||g^x)), V_C, x\}$ and spends it by computing $P_C = g^{V_C}$ and $G = g^x$, and sending $\text{Sign}_S(\mathcal{H}(P_C||G))$, P_C, and G to the server. Then, employing the secret key V_C the app signs a bitstring representing the current time and sends the resulting signature $\text{Sign}_C(\mathcal{H}(CurrentTime))$ to the server.
2. The server verifies the RSA signature $\text{Sign}_S(\mathcal{H}(P_C||G))$ using its public key P_S, the signature $\text{Sign}_C(\mathcal{H}(CurrentTime))$ under the e-coin public key P_C and checks that the e-coin has not been spent before.
3. The app determines the current time slot, $Slot$. Then it generates $\mathtt{N}$ random keys $\{K_i\}_{0\leq i<\mathtt{N}}$, and computes the set $\{ID_{Slot i} = \text{HMAC}_{K_i}(Slot||License)\}_{0\leq i<\mathtt{N}}$. It also generates $\mathtt{N}$ even numbers $\{x_{ri}\}_{0\leq i<\mathtt{N}} \in [0, (q-1)/2)]$ and computes $\{G'_i = g^{x'_i} = g^{x+x_{ri}}\}_{0\leq i<\mathtt{N}}$.
4. The app next computes the following $\mathtt{N}$ hash digests $\{\mathcal{H}(Slot||ID_{Slot i}||G'_i)\}_{0\leq i<\mathtt{N}}$, blinds each of them with a different blinding factor r_i, and sends the blinded results to the server.
5. The server randomly chooses an index $j \in [0, \mathtt{N}-1]$ and sends it to the app.
6. For each $i \neq j$, the app sends r_i, x_{ri}, $Slot$ and $ID_{Slot i}$ to the server.
7. The server unblinds the $\mathtt{N}-1$ hash digests H_i, computes $G'_i = G \cdot g^{x_{ri}}$, and checks whether each H_i is equal to $\mathcal{H}(Slot||ID_{Slot i}||G'_i)$. It also verifies that each x_{ri} is even and falls in the $[0, (q-1)/2)]$ range, and checks that $Slot$ corresponds to a future time slot.
8. If all the checks are satisfied, the server blindly signs the remaining blinded hash and returns the result to the app. Also, the server stores:

$$\{P_C, G, \text{Sign}_S(\mathcal{H}(P_C||G)), CurrentTime, \text{Sign}_C(\mathcal{H}(CurrentTime))\}.$$

After running this procedure, the app obtains the server signature on the ticket,

namely $\text{Sign}_S(\mathcal{H}(Slot||ID_{Slot}||g^{x'}))$. If $x' \in [0, q-1]$ is even, the ticket is valued. For each *valued* ticket, the app stores:

$$\{\text{Sign}_S(\mathcal{H}(Slot||ID_{Slot}||g^{x'})), Slot, K, x'\}.$$

Remark. Whether or not a ticket is valued depends on the parity of x'. The system server must check that x (in the e-coin) and x' (on the ticket) have the same parity. Since $x' = x + x_{r_j}$, this is achieved by checking that x_{r_j} is even by means of the described cut-and-choose technique.

Parking inspection

A parking officer approaches the car and queries its on-board device.

1. The on-board device, from its internal clock, determines the current time slot, *Slot*, and sends it to the parking officer together with the car plate number, *License*, and the signature $\text{Sign}_D(\mathcal{H}(Slot||License))$.
2. The parking officer checks the received *Slot* and *License* values and sends them together with $\text{Sign}_D(\mathcal{H}(Slot||License))$ to the system server.
3. The server sends a push message to the app associated with *License* and waits for a valid ticket (see the "Ticket request" procedure) for some time.
4. If the app does not properly respond to the request, the car is fined and its driver receives a telematic notification.

Remark. Possession of $\text{Sign}_D(\mathcal{H}(Slot||License))$ proves that the parking officer is located close to the car.

Ticket request

When the app is requested to prove a pending inspection, the following procedure is run:

1. The app:
   (a) Connects to the server and asks for the signature computed by the on-board device (during the "Parking inspection" execution) over *Slot* and *License*.
   (b) Verifies the signature $\text{Sign}_D(\mathcal{H}(Slot||License))$.
   (c) If the previous verification is successful, it sends the ticket $\text{Sign}_S(\mathcal{H}(Slot||ID_{Slot}||g^{x'}))$, *Slot*, ID_{Slot}, K, and x' to the server.
2. The system server:
   (a) Checks the signature $\text{Sign}_S(\mathcal{H}(Slot||ID_{Slot}||g^{x'}))$ over *Slot*, ID_{Slot}, and $g^{x'}$ under the server public key P_S.
   (b) Verifies that ID_{Slot} equals $\text{HMAC}_K(Slot||License)$.
   (c) Checks that $x' \in [0, q-1]$ is even and has not been previously revoked. In such a case, the timestamp issued during the revocation, $\text{Timestamp}_{TSA}(x')$, is returned (see the "Ticket revocation" procedure).
3. If all the previous checks are satisfied, the parking officer is informed about the validity of the ticket provided.

If the app cannot provide a valid ticket, the driver is fined. If the driver paid for parking but her phone is out of coverage, she will also be fined. In that case, the fine will be canceled when the mobile device provides the ticket after reconnecting.

Ticket revocation

If the driver removes the car earlier than expected, she can revoke unused valued tickets and recover the paid amount. The app connects to the server through an anonymous channel, and, for each valued ticket:

1. The mobile app sends $\text{Sign}_S(\mathcal{H}(Slot||ID_{Slot}||g^{x'}))$, $Slot$, ID_{Slot} and x'.
2. The system server:
   (a) Validates the signature $\text{Sign}_S(\mathcal{H}(Slot||ID_{Slot}||g^{x'}))$ over $Slot$, ID_{Slot}, and $g^{x'}$ under the public key P_S.
   (b) Checks that $Slot$ corresponds to a future time slot.
   (c) Checks that x' is even and has not been revoked before. In such a case, its $\text{Timestamp}_{TSA}(x')$ would be returned as proof.
3. If the ticket to be revoked is valid it requests the TSA to timestamp x'. It stores x' and $\text{Timestamp}_{TSA}(x')$ in a database.
4. Finally, a new valued e-coin is issued and stored by the app wallet (by running the "E-coin request" procedure).

Remark. Before revoking a ticket, the system server checks that it is valued (x' is even). The timestamp on x' will serve, in case of dispute, as proof of its revoked status.

3.5 Privacy and security analysis

This section describes the privacy analysis and the security analysis against a malicious driver.

3.5.1 Privacy analysis

When a driver parks, her app requests a constant amount of tickets through the "Ticket generation" procedure. At this moment, the system server knows that a parking operation is just being begun, but it must be unable to determine: (1) The plate number of the car; (2) Which tickets are valued and which are not (so that the expected duration remains secret).

When a parking officer checks the parking status of a car, the driver's app is asked to provide the ticket corresponding to the current time slot ("Ticket request" procedure). At this moment, the exact time at which the provided ticket was issued should not be revealed (so that the parking starting time remains secret).

These objectives are addressed in the following lemmas and theorem.

Lemma 3.5.1. *The information collected by the system server during the issuance of a ticket ("Ticket generation" procedure) cannot be linked to the car license plate number.*

Proof. When an e-coin is issued ("E-coin request" procedure), the server blindly signs it, so that it obtains no information about it. Hence, when it is spent, it cannot be linked to the moment it was generated or to the driver who requested it.

During the issuance of a ticket ("Ticket request" procedure), the relationship between the ticket and a license plate number is established when computing each value ID_{Sloti} (step 3) as $\text{HMAC}_{K_i}(Slot||License)$.

As a result of the cut-and-choose checks (step 6), the app sends x_{r_i}, $Slot$, and ID_{Slot_i} to the server. The value of x_{r_i} has been set at random so that there is no relation between it and the car plate number. $Slot$ is just an integer representing a time slot. Regarding ID_{Slot_i}, it has been computed as $\text{HMAC}_{K_i}(Slot||License)$ so that the relation between ID_{Slot_i} and $License$ can only be determined if K_i is known. Since the app does not reveal it, a brute force search would have to be performed which is assumed to be unfeasible. □

Lemma 3.5.2. *When a ticket is generated, the information collected by the system server does not allow it to be determined whether or not the ticket is valued.*

Proof. When a ticket is issued ("Ticket request" procedure), the app spends an e-coin by sending $\text{Sign}_S(\mathcal{H}(P_C||G))$, P_C, and G (step 1), with $G = g^x$. Whether or not the spent e-coin is valued depends on the parity of x. Since the app does not reveal x, an attacker would have to solve an instance of the discrete logarithm problem which is assumed to be unfeasible. □

Lemma 3.5.3. *The information provided by a driver when a ticket is requested at a given time slot cannot be linked to other time slots.*

Proof. When a ticket is requested, the app sends $\text{Sign}_S(\mathcal{H}(Slot||ID_{Slot}||g^{x'}))$ with $Slot$, ID_{Slot}, K and x' to the system server.

The relationship between that ticket and the spent e-coin is given by $g^{x'} = G \cdot g^{x_r}$ (step 3), x_r being an even number. Since x_r is never revealed, the relationship mentioned cannot be determined. Since tickets are generated independently, there is not any information relating them. Hence, a ticket cannot be related either to the e-coin spent during its issuance or to other tickets.

□

Theorem 3.5.4. *The protocol fulfills the objectives defined in Section 3.3.3.*

Proof. Lemma 3.5.1 states that the information collected by the server as a result of ticket issuance cannot be linked to cars. Also, during a ticket request, the app only provides a ticket if the server sends a message signed by the on-board device in the car. Such a signed message can only be obtained if a parking officer is close to the car. These signed messages include the current time slot, so that they cannot be employed to obtain tickets for other slots. Hence, objective 1 is fulfilled.

Regarding objective 2, when the app provides a ticket, that ticket only contains information about the requested time slot. Lemma 3.5.3 states that a ticket cannot be related to other tickets. Also, Lemma 3.5.2 proves that the expected duration of a parking operation is not leaked during the issuance of tickets. Hence, objective 2 is met.

Since the tickets issued for all the time slots composing the expected parking duration are generated at the beginning of the parking operation, an Internet connection is only required at that moment. If a parking inspection is performed while the app is out of coverage, the app is allowed to send the ticket later. Hence, objective 3 is also met.

□

3.5.2 Security analysis

We first prove two basic properties of e-coin systems: *unforgeability* and *security against double spending*. Next, we will deal with two attacks that a malicious driver might attempt to perform.

Lemma 3.5.5. *The proposed e-coin system is* unforgeable *and* secure against double spending.

Proof. An e-coin is only valid if its header digest, $\mathcal{H}(P_C||g^x)$, has been signed by the server. Since RSA signatures over hashed messages are unforgeable, so are the e-coins. When the server accepts a payment, it stores the received e-coin in a database. Before accepting an e-coin, the server checks that the received e-coin has not been spent before.

□

Generation of free-of-charge valued e-coins

A malicious driver could obtain free-of-charge valued e-coins by running a request for no-valued ones but setting an even value for x (step 3.4.2 of the "E-coin request" procedure). It would have to generate the set $\{x_i\}_{0\leq i<\mathtt{N}}$ so that all its elements but one are odd. Then it would need to be lucky so that the server, during the cut-and-choose checking, chooses an index j corresponding to the only even number in the set. This attack succeeds with probability $1/\mathtt{N}$.

The aforementioned attack can be avoided by blocking drivers that have failed the cut-and-choose checking a certain amount of times.

Generation of valued tickets from no-valued e-coins

A driver could cheat, during the issuance of a ticket ("Ticket generation" procedure), by obtaining a valued ticket after spending a no-valued e-coin. This attack requires that, at Step 3, the driver generates the set $\{x_{r_i}\}_{0\leq i<\mathtt{N}}$ so that all its elements but one are even, and the server, during the cut-and-choose checking, chooses index j so that x_{r_j} is the odd element. The probability of success for a given ticket is $1/\mathtt{N}$.

Since tickets are requested anonymously, the author of such an attack cannot be identified and blocked. We propose that all the R tickets of a given parking operation be

requested in batch so that the server sends all the challenges for the R cut-and-choose checks at the same time. The server will only sign the requested tickets if all the cut-and-choose checks are satisfied. If some of them fail, the tickets are not signed but the employed e-coins are stored as being spent.

Under these conditions, the best strategy for an attacker consists of requesting all the R tickets employing no-valued e-coins and cheating by trying to obtain just one valued ticket. If the attacker succeeds, they will obtain one free valued ticket. Otherwise, they will lose all the R no-valued e-coins. The probability of success is $1/\mathtt{N}$.

Since no-valued e-coins are spent, we can prevent the attack from being launched massively by limiting their generation. We propose limiting the amount of no-valued e-coins a driver can obtain so that for each valued e-coin, she can request only up to $R - 1$ no-valued ones. This rate still allows users to make parking operations involving just one time slot.

When the attacker succeeds, they obtain a ticket after spending R no-valued e-coins. Since the probability of success is $1/\mathtt{N}$, they spend, on average, $\mathtt{N} \cdot R$ no-valued e-coins for each free ticket. Obtaining such an amount of no-valued e-coins needed a request for $\frac{N \cdot R}{R-1} \approx \mathtt{N}$ valued e-coins which have been paid.

We conclude that the mentioned attack is possible but, on average, the attacker has to pay for N valued e-coins, for each free ticket they obtain. Hence, the benefit for an attacker is really negligible.

3.6 Experimental results

The two main actors of the system are the server and the mobile application. The mobile application has been implemented in `Java` for Android devices. The big number library `java.math.BigInteger` has been used to implement cryptographic operations involving big integers. The server part has been developed under the `.Net` framework.

The feasibility of the system has been analyzed by testing the time-consuming procedures, namely "E-coin request" and "Ticket generation" which are the procedures involving cut-and-choose checks. The remaining procedures are rarely run or have a much lower cost.

Our implementation employs 2048 bits cryptography for RSA/DSA signatures and for all the operations involving hard-to-solve instances of the discrete logarithm problem. Hash digests are computed by means of the SHA-256 function.

We have measured the average time to generate a valued e-coin, a no-valued e-coin and a ticket for both a serial and a parallel implementation. As expected, mobile phones and computers with more powerful CPU's achieve better performances. We have also observed that increasing the number of cores in parallel executions reduces the running times. Memory requirements are negligible.

The cut-and-choose checks have been implemented with $\mathtt{N} = 100$. For that reason, the generation of no-valued e-coins and tickets is approximately 100 times harder than the generation of valued e-coins.

Table 3.2 shows the average running times (in milliseconds) obtained over two mobile phones. Table 3.3 shows the running times at the server part.

Mobile phone			**Valued e-coin**		**No-valued e-coin**		**Ticket**	
BQ Model	Cores	GHz	Serial	Parallel	Serial	Parallel	Serial	Parallel
Aquaris U Lite	4	1.4	96	38	8998	3368	4646	1741
Aquaris U	8	1.4	92	35	8976	2248	4643	1164

Table 3.2: Mobile application running times (in milliseconds)

A 4-Core (8-Thread) server requires approximately 23 seconds to process a complete parking transaction (involving the generation of 12 tickets). A more powerful computer, namely a 36-Core (72-Thread) professional server, would reduce that time to around 2.5 seconds.

Server				**Valued e-coin**		**No-valued e-coin**		**Ticket**	
Processor	Cores	Threads	GHz	Serial	Parallel	Serial	Parallel	Serial	Parallel
i3	2	4	3.40	125	42	10957	4148	11296	4277
i5	2	4	3.00	141	51	12372	5038	12661	5575
Athlon	4	4	2.80	157	42	15483	3925	15927	4266
i7	4	8	3.40	124	31	9001	1840	9297	1901

Table 3.3: Server running times (in milliseconds)

Reloading the electronic wallet with 12 e-coins, 6 of them being valued, took 13 seconds by the mobile phone and 12 seconds by the server. If 9 out of the 12 requested coins are valued, the times reduce to 6 and 7 seconds, respectively. The overall time for the issuance of 12 tickets is 37 seconds, 14 of them corresponding to operations computed by the mobile phone while the remaining 23 are for computations by the server. The app computations to request an e-coin could be pre-computed so that the delay in e-coin issuance could be reduced to that at the server part.

Chapter 4

Reusability of e-tickets

This chapter was summarized in the scientific paper *"A Construction for Providing Reusability to Mobile Phone-Based e-Tickets"* published in the *IEEE Access* journal (2020) [BS20a]. The chapter proposes a construction that can be coupled to an existing privacy-enabled e-ticketing system and endows it with the reusability property.

4.1 Introduction

In 1662, the first bus service was deployed in Paris with seven horse-drawn vehicles running along regular routes [19b]. Since then, public transport systems have evolved continuously. In the digital era, the worldwide deployment of the Internet together with the popularization of mobile phones has set the basis for the development of advanced technology for the management of public transport. As a very relevant example, we mention the replacement of paper tickets with electronic ones (e-tickets).

From the data privacy point of view, an accurate analysis of data about the particular way in which a person uses public transport allows personal information to be inferred. So as to reduce such risks, companies should guarantee that the information they collect is restricted to that strictly necessary to provide the service requested by their customers. In the particular case of public transport e-tickets, the information managed by a ticket issuer must not allow the way a specific citizen uses public transport to be traced.

4.1.1 E-ticket systems

A *ticket* is defined as *"a small piece of paper or card given to someone, usually to show that they have paid for an event, journey, or activity"* [19a]. The definition for *e-ticket* is *"a ticket, usually for someone to travel on an aircraft, that is held on a computer and is not printed on paper"* [19a].

Many e-ticket systems [Fuj+99; Ver+08; QH05; Wan+04a; LQP09] consider a system model composed of three actors in which *users* make use of a service offered by a *service provider* through an e-ticket generated and managed by the *ticket issuer*. The service provider and ticket issuer roles may be taken by the same entity.

Table 4.1: Security and privacy requirements for e-ticket systems

Requirement	**Description**
Guarantee	
Integrity	An e-ticket cannot be modified.
Authenticity	A user must be able to validate that an e-ticket has been issued through an authorized provider.
Non repudiation	The service provider cannot deny having issued an e-ticket.
Unforgeability	Only authorized entities can issue a valid e-ticket.
Non overspending	An e-ticket can only be used the number of times stipulated at the time of its issuance.
Exculpability	The service provider cannot falsely accuse an honest user.
Consider	
Identifiable e-ticket	The identity of the e-ticket owner can be verified.
Anonymity	
Fully-Revocable	Anonymity of the ticket owner can be lifted.
Selective-Revocable	The identity of the ticket owner can be revealed in case of a malicious act.
Anonymous	The identity of ticket owners cannot be determined in any way.
Transferability	An e-ticket can be transferred to another user.
Reusability	An e-ticket can be used for several journeys.

There exist proposals in which e-tickets can be used anonymously [Arf+15; Rup+13; Arf+14; AKJ10; AAA17; QH05; Ver+08; Wan+04a; Viv+12]. Some of them [Arf+15; Arf+14; Ver+08; Wan+04a; Viv+12] include a *trusted third-party* responsible for managing anonymity, whereas in other proposals [Rup+13; AKJ10; AAA17; QH05] anonymity is managed by the ticket issuer itself. Additionally, there may exist the so-called *inspection authorities* responsible for face-to-face checks [Mil+12].

Most proposals agree upon the existence of the following three main phases during the lifetime of an e-ticket [Mut+12]:

1. **Payment.** The user pays for the required service.
2. **Issuance.** The ticket issuer, after receiving an appropriate payment, generates an e-ticket that is provided to the user.
3. **Validation.** The service provider verifies the validity of the e-ticket provided before granting the user with access to the service.

Some proposals consider *payment* and *issuance* as being part of the same phase [PC97; SVW08; AKJ10; Rup+13], whereas others include an *anonymity revocation* procedure which allows the identity of users who have acted fraudulently to be determined [Arf+15;

Rup+13; Arf+14; AKJ10; AAA17; QH05; Ver+08]. An additional *random inspection* process performed by inspection authorities is also considered in some proposals [Mil+12].

An e-ticket system must guarantee, or at least consider, the security and privacy requirements enumerated in Table 4.1 [Mut+12]. *Reusability* allows an e-ticket to be used for several journeys during a period of time. Throughout this chapter, the word *journey* will usually refer to an *elementary journey* in which a user moves from a starting to a destination station with no transfers. Reusability is especially useful in cities providing several means of transport such as bus, tram, train, or underground. Citizens and tourists benefit from great flexibility to plan their journeys inside the city.

The system must allow it to be checked that an e-ticket has not expired, or that its maximum amount of elementary journeys has not been exceeded. In both cases, fraudulent uses must be detectable [Mut+12].

4.2 Related work

There are plenty of proposals for e-ticket systems in the literature. Most of them do not consider the possibility of carrying out transfers [Mil+12; PC97; Han+17; Arf+15; Rup+13; KLG13; SVW08; AKJ10; Viv+11; AAA17], but implement mechanisms allowing clients to purchase and validate e-tickets anonymously. Specifically, the proposals [Viv+11; AAA17; SVW08] make use of pseudonyms so that the system cannot link an e-ticket to the identity of the user who acquired it. If pseudonyms are used, the real identity information is not included in the ticket, only its pseudonym. To avoid user traceability, pseudonyms should be updated regularly. Other proposals provide anonymity by means of alternative techniques [Arf+15; Rup+13; AKJ10; PC97; Han+17; KLG13]. For example, in an attribute-based credentials scheme, a user can prove that he has obtained a credential without revealing any additional information.

In case of fraudulent behavior, some proposals allow anonymity revocation [Arf+15; Rup+13; AKJ10; AAA17]. Anonymity has to be revocable in order to identify fraudulent users while honest ones must remain anonymous. The real identity of fraudulent users is usually revealed by a trusted third party.

The proposal in [Mil+12] is the only one including the possibility of carrying out random inspections. An inspector can require a user to prove that she is the owner of the inspected e-ticket. At this moment the system is able to link the identity of the user to the e-ticket and obtain all the information about her journey (like the starting station).

There exist a few proposals allowing transfers so that the rider of a public transport vehicle can continue the trip on another bus or train [Hey+06; QH05; Viv+12]. In particular [QH05; Viv+12] include revocable anonymity. They were both designed to be implemented on mobile devices. Some proposals limit the amount of transfers, whereas others allow an unlimited amount within a certain period of time.

Systems limiting the number of transfers usually implement a mechanism in which an e-ticket includes a hash chain whose size depends on the maximum number of transfers allowed. In this way, if $\mathtt{T}$ journeys are allowed, a $\mathtt{T}$-element hash chain $\{chain_k\}_{1 \leq k \leq \mathtt{T}}$ is

Table 4.2: Summary related work proposals

Proposal	Allow transfers	Expiration time	Limited transfers	Anonymity		Anonymity revocable	Allow inspections
				Pseudonym	Others		
[Mil+12]	✗	✗	✗		✓	✗	✓
[PC97]	✗	✗	✗		✓	✗	✗
[Han+17]	✗	✗	✗		✓	✗	✗
[Arf+15]	✗	✗	✗		✓	✓	✗
[Rup+13]	✗	✗	✗		✓	✓	✗
[KLG13]	✗	✗	✗		✓	✗	✗
[SVW08]	✗	✗	✗	✓		✗	✗
[AKJ10]	✗	✗	✗		✓	✓	✗
[Viv+11]	✗	✗	✗	✓		✗	✗
[AAA17]	✗	✗	✗	✓		✓	✗
[Hey+06]	✓	✓	✗	✓		✓	✗
[QH05]	✓	✓	✓	✓		✓	✗
[Viv+12]	✓	✓	✓	✓		✓	✗
[BS20a]	✓	✓	✓	*Determined by the underlying e-ticket system*			✓

generated so that $chain_{k-1} = \mathcal{H}(chain_k)$, with $chain_{\mathrm{T}}$ being a random number. When beginning the i-th journey, the user is asked to provide $chain_i$.

Table 4.2 presents a summary of the properties fulfilled by the cited previous proposals.

Finally, to the best of our knowledge, no previous proposal allowing a limited amount of transfers includes the possibility of carrying out random inspections. In such a case, when a user identifies herself and proves that she is the owner of the e-ticket, the system is able to link her identity to all the journeys made with the inspected ticket. The system should only be able to determine whether the inspected user is authorized to make the current journey with the ticket provided.

4.3 System model

The system model (Figure 4.1) is composed of the following actors:

1. **Mobile application** or **app.** It runs on the mobile phone of public transport users. It is used for e-ticket purchase and validation. Each time a user enters a means of transport (after an e-ticket validation or a transfer), the mobile phone is to be placed close to a hardware device which will grant access. When a user leaves, a similar operation is necessary to obtain the data needed for the next journey in case a transfer is to be carried out. If a random inspection takes place, the user will use the app to demonstrate the validity of her ticket. The app also manages the so-called *deposit token*, employed to force all users to run the "Get-out" procedure at the end of each journey, even when they are not going to do a transfer.
2. **System server.** This is a central server which manages all the information of the e-ticket system. It is contacted by the users' app during the purchase and validation of e-tickets. It is also accessed by transport system proximity devices

when users run the "Get-in" and "Get-out" procedures, including the management of *deposit tokens*. Inspectors also access it when performing random inspections. It incorporates a database to store data and avoid double spending frauds.

3. **Transport system devices.** Our proposal requires the transport system provider to install a proximity reading device at each entrance and exit.
   - An **entrance device** is accessed by users when entering a transport system. It is responsible for validating whether the data provided by a user (a *travel token*) allow her to enter the transport system. In that case, it grants access to the user and provides her with data (an *inspection token*) that will be requested if an inspection takes place.
   - An **exit device** is accessed by users when leaving a transport system. This access is mandatory for all users, even if they are not going to do a transfer. If not done, the user loses her *deposit token*.

   Both devices need permanent communication with the system server.
4. **Inspector.** He is responsible for carrying out random controls within the transport system. He verifies that users are traveling with a valid *inspection token*. The inspector uses a mobile device to that end.

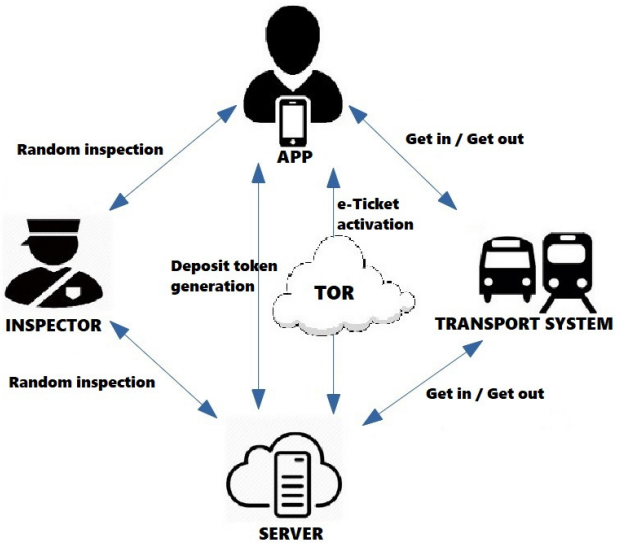

Figure 4.1: System model

4.4 System proposal

4.4.1 System overview

Our proposal is designed as a *plug-in* to be coupled to an existing e-ticket system. The underlying e-ticket system has its own purchase and anonymity policies. When a user

starts a new journey (maybe composed of several stages with the corresponding transfers), she must activate an available e-ticket on the underlying system. Just after that, the "Travel token chain generation" procedure is run. During its execution, the *plug-in* generates a *travel token* chain whose length equals the maximum amount of journey stages allowed per e-ticket. When this procedure concludes, the first *travel token* of the chain is *ready to be used*. These chained *travel tokens* are to be presented sequentially at the beginning of each elementary journey. Figure 4.2 is an abstract graphic of our proposal.

Just before gaining access to a transport system vehicle or platform, the user performs a "Get-in" procedure during which she is required to hold her mobile phone near to an entrance control device and provide a *travel token* via some proximity communication system. The entrance device, by querying the central system server, determines whether the *travel token* provided is *ready to be used* and checks that it has not been used before. If the validation is correct, the entrance device will generate, and send to the mobile device, an *inspection token* which includes a *trip identifier*. A *trip identifier* is a random value which uniquely identifies each trip carried out by the transport system. The *inspection token* will be required in case of inspection.

When a user leaves a platform or vehicle, she runs the "Get-out" procedure. She must then hold her mobile phone close to an exit device in order to obtain her next *travel token ready to be used* for the next stage of the journey.

Figure 4.2: Abstract graphic

During a "Random inspection", the user must provide the *inspection token* she obtained when performing the "Get-in" procedure. That *inspection token* is linked to some piece of personal information, such as her passport number or the hash digest of a personal picture, which will be checked by the inspector.

To avoid fraudulent use, our system requires all users to perform the "Get-out" procedure at the end of each stage of the journey, even when they are not going to do a transfer. This is achieved by making users provide a *deposit token* during the execution of the "Get-in" procedure and obtaining a new one free of charge when performing the corresponding "Get-out" procedure. If this is not done, the user loses her deposit and will have to pay for a new one. The first *deposit token* of each user is provided free of charge.

4.4.2 Preliminary set-up procedures

Next we describe two procedures related to system configuration. The first one, "Server set-up", is run by the ticket issuer to create and set the cryptographic keys and constants required for the system to run. The other one, "Application set-up", is run by users when installing the system app in their mobile devices.

Server set-up

Before deploying the system, the ticket issuer:

1. Creates the following cryptographic keys:
   - "Get-in" blind signature (Section 2.5) key pair: $P_{S^{in}}/V_{S^{in}}$.
   - "Get-out" blind signature key pair: $P_{S^{out}}/V_{S^{out}}$.
   - "Deposit" blind signature key pair: P_{S^d}/V_{S^d}.
   - "User token" blind signature key pair: P_{S^u}/V_{S^u}.
   - "Trip" partially blind signature (Section 2.6) key pair: P_{S^t}/V_{S^t} .

   The public keys created are: $P_{S^{in}}, P_{S^{out}}, P_{S^d}, P_{S^u}, P_{S^t}$.
   The private keys created are: $V_{S^{in}}, V_{S^{out}}, V_{S^d}, V_{S^u}, V_{S^t}$.
2. Creates a tuple of set-up parameters for a commitment-scheme allowing range zero-knowledge proofs (Section 2.8).
3. Sets and stores the following constants:
   - $\texttt{T} \equiv$ Maximum amount of journeys allowed per e-ticket.
   - $\texttt{TIME\_ACTIVE} \equiv$ Amount of minutes for which an e-ticket is valid after its activation.
   - $\texttt{TT}_{\texttt{end}} \equiv$ A hash on random string.
   - $\texttt{N} \equiv$ Parameter which tunes the security level of the cut-and-choose checking performed during the generation of *travel token* chains.

The private keys $V_{S^{in}}, V_{S^{out}}, V_{S^d}, V_{S^u}, V_{S^t}$ must be kept secret by the ticket issuer which will make them available only to the system server. The remaining parameters are made public to all the actors.

Application set-up

A user has to install the application of the underlying e-ticket system together with the extension *plug-in* on her mobile phone. Then, she must introduce all the information required by the underlying e-ticket system (credit card, etc). After that, the extension *plug-in* will request some information that identifies her unequivocally (like a picture or her passport number) which will be stored in the ID_{user} object in her phone. This information must allow an inspector to verify her identity.

The app obtains a *deposit token* free of charge by running the procedure described in Section 4.4.4.

4.4.3 *Travel token* chain

After a ticket is activated, the system generates a chain composed of T chained *travel tokens*, $\{TT_i\}_{0 \leq i < \mathtt{T}}$, T being the maximum amount of journeys (with the corresponding transfers) that can be made with each e-ticket. Each *travel token* in a given chain is assigned the same expiration time E and is linked to the same masked user identity ID_{masked} (computed as $\mathrm{HMAC}_{K_u}(ID_{user})$ with K_u being a secret key).

Structure of a travel token

A *travel token* (Figure 4.3) is a tuple containing several public parameters computed from a set of secret ones. The private parameters are generated and kept secret by the app. The public ones compose the *travel token* object, TT_i.

***Travel token* i**
Private K_i, u_i, c_i, t_i
Public · $\mathrm{Blind}_{u_i}(\mathrm{Mask}_{K_i}(User))$ · $\mathrm{Commit}_{c_i}(Expiration\ time)$ · $\mathrm{Blind}_{t_i}(Travel\ token\ i+1)$

Figure 4.3: The i-th *travel token*

- Private parameters of TT_i:
  - K_i: secret key used to re-mask the masked user identity.
  - u_i: secret value for blinding the re-masked user identity.
  - c_i: secret value for committing to the expiration time.
  - t_i: secret value for blinding the link to the next *travel token* of the chain.
- Public parameters of TT_i:
  - U_i: blinded re-masked user identity, namely $\mathrm{Blind}_{u_i}(\mathcal{H}(\mathrm{HMAC}_{K_i}(ID_{masked})), P_{S^u})$.
  - C_i: commitment to the expiration time E, namely $\mathrm{Commit}_{c_i}(E)$.
  - $NextTT_i$: blinded link to the next *travel token* of the chain, namely $\mathrm{Blind}_{t_i}(\mathcal{H}(TT_{i+1}), P_{S^{in}})$.
- The chain is concluded by setting $TT_{\mathtt{T}} = \mathtt{TT}_{\mathtt{end}}$.

A *travel token* TT_i is said to be *ready for use* when it is accompanied by two digital signatures under the "Get-in" and "Get-out" server key pairs, namely $\mathrm{Sign}_{S^{in}}(\mathcal{H}(TT_i))$ and $\mathrm{Sign}_{S^{out}}(\mathcal{H}(TT_i))$. A *ready for use travel token* allows a user to obtain access to a transport system.

Generation of a *travel token* chain

A *travel token* chain (Figure 4.4) is generated by means of the following procedure which takes as input and expiration time E and a masked user identity ID_{masked}:

1. Set $i = \mathtt{T} - 1$
2. While $i \geq 0$ (generate TT_i):

- Generate K_i, u_i, c_i, and t_i at random and store them secretly.
- Set $U_i = \text{Blind}_{u_i}(\mathcal{H}(\text{HMAC}_{K_i}(ID_{masked})), P_{S^u})$.
- Compute $C_i = \text{Commit}_{c_i}(E)$.
- Compute $NextTT_i$:
  - if $i < \mathtt{T} - 1$ then
    $NextTT_i = \text{Blind}_{t_i}(\mathcal{H}(TT_{i+1}), P_{S^{in}})$.
  - if $i = \mathtt{T} - 1$ then
    $NextTT_i = \text{Blind}_{t_i}(\mathcal{H}(\mathtt{TT_{end}}), P_{S^{in}})$.
- Pack U_i, C_i and $NextTT_i$ into a tuple and name it TT_i.
- Let $i = i - 1$.

3. The resulting *travel token* chain is represented as a set $\mathcal{C} = \{TT_0, \ldots, TT_{\mathtt{T}-1}\}$.

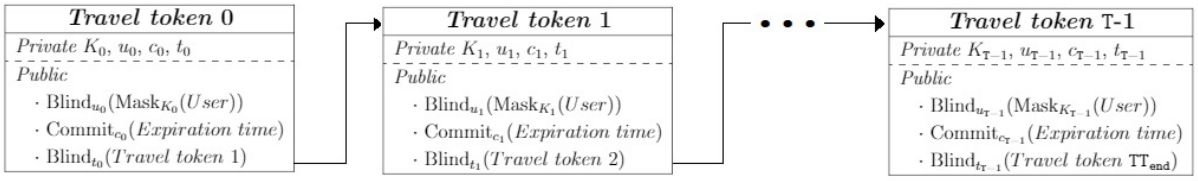

Figure 4.4: *Travel token* chain

Validation of a *travel token* chain

Given $\mathcal{H}(TT_0)$, the correct composition of a *travel token* chain can be checked by requiring its generator to provide the input parameters E and ID_{masked} together with all of its private and public parameters. The verifier can then repeat the construction process from $TT_{\mathtt{T}-1}$ down to TT_0 and finally check that the hash of the obtained token TT_0 matches $\mathcal{H}(TT_0)$.

4.4.4 Travel-related procedures

The following section describes the protocols executed by users while traveling in public transport.

Deposit token generation

When a user enters a transport system, she has to provide a *deposit token*. Later, when she leaves, she receives a new one which is generated by means of the process described in this section. As a result of this process, the user obtains a new *deposit token* whose structure is that of an anonymous e-coin generated through Chaum's system [Cha83].

A *deposit token* is generated free of charge the first time a user registers with the system or when obtained during the "Get-out" process. If a user loses her *deposit token*, probably because she does not run the "Get-out" procedure at the end of a journey, she will be charged to obtain a new one.

A *deposit token* is generated as follows:

1. If necessary, the app performs a credit card payment for the cost of a *deposit token*.
2. The app generates a random value *deposit*, hashes and blinds it with a random blinding factor d for server's deposit key, $\text{Blind}_d(\mathcal{H}(deposit), P_{S^d})$, and sends the resulting value to the server.
3. The server blindly signs the value received and returns the resulting blinded signature to the app.
4. The app unblinds the blinded signature received, obtaining a signature from the server on $\mathcal{H}(deposit)$, namely $\text{Sign}_{S^d}(\mathcal{H}(deposit))$.
5. Finally, the app stores the resulting *deposit token* tuple $\{deposit, \text{Sign}_{S^d}(\mathcal{H}(deposit))\}$.

When the previous procedure is executed during a "Get-out" procedure (step 1 is skipped), the identity of the user is not revealed. When a payment is made, anonymity is not required since the identity of the user can be obtained from the payment method data. This case of usage shall not be applied to users acting as required.

E-ticket purchase

The user pays for and obtains e-tickets from the underlying e-ticket system. The underlying e-ticket system is required to guarantee that the purchased e-tickets can later be activated anonymously.

E-Ticket activation

When a user is about to start a journey (which may include several transfers), her app contacts the ticket issuer server and activates an e-ticket from the underlying system. After a successful activation, the *plug-in* module runs the following "Travel token chain generation" procedure.

Travel token chain generation. This procedure generates a chain composed of `T` *travel tokens*. Its correct composition is checked by means of a cut-and-choose technique tuned with parameter `N`.

1. The app sets the expiration time to $E = CurrentTime + \texttt{TIME\_ACTIVE}$.
2. The app takes ID_{user} and computes a set of `N` masked identities $\{ID_{masked_k} = \text{HMAC}_{K_{U_k}}(ID_{user})\}_{0 \leq k < \texttt{N}}$ from `N` different keys K_{U_k}.
3. Using the procedure in Section 4.4.3, the app generates `N` *travel token* chains $\{\mathcal{C}^k\}_{0 \leq k < \texttt{N}}$ providing E and ID_{masked_k} as input, respectively.
4. The app, for each chain $\mathcal{C}^k = \{TT_0^k, \ldots, TT_{\texttt{T}-1}^k\}$, hashes its first *travel token*, TT_0^k, and blinds the resulting digest under a random blinding factor r_k generating a set of blinded digests $\{B_k = \text{Blind}_{r_k}(\mathcal{H}(TT_0^k), P_{S^{in}})\}_{0 \leq k < \texttt{N}}$. All these blinded values are sent to the server.
5. The server chooses a random $j \in [0, \texttt{N} - 1]$ and sends it to the app.
6. The app, for each $k \neq j$, sends the blinding factor r_k to the server together with all the information required to check the correct composition of chain $\mathcal{C}^k$.

7. For each $k \neq j$, the server unblinds the received hash and verifies each chain C^k under $\mathcal{H}(TT_0^k)$, E and ID_{masked_k} together with all the additional parameters as described in Section 4.4.3.
8. If all the validations are satisfied, the server blindly signs B_j under its "Get-in" key pair so that the app is able to obtain $\text{Sign}_{S^{in}}(\mathcal{H}(TT_0^j))$.
9. Next, the app generates r'_j at random and computes $\text{Blind}_{r'_j}(\mathcal{H}(TT_0^j), P_{S^{out}})$ which is sent to the server which will blindly sign it under its "Get-out" key pair. From the result, the app obtains the signature $\text{Sign}_{S^{out}}(\mathcal{H}(TT_0^j))$.
10. Finally, the app keeps a chain of *travel tokens* $\mathcal{C}^j = \{TT_0^j, \ldots, TT_{\mathtt{T}-1}^j\}$, whose first token, TT_0^j has been blindly signed by the server under the "Get-in" and "Get-out" key pairs. Hence, this first *travel token* is *ready for use*.

If any check fails, the procedure of the underlying e-ticket system applied in case of fraudulent behaviour is run. This may include anonymity revocation and fining the user. Although the probability of cheating cannot be made arbitrarily small (making $1/N$ negligible would require a rather large value for $\mathtt{N}$ which has a direct impact on the running time of the protocol), users will be discouraged from cheating if the amount of the fine is large enough.

The chain generation process ensures that all the *travel tokens* of a chain are linked to the same ID_{user} object so that they cannot be shared among different users. The user linked to the *travel tokens* of the chain shall not necessarily be the one who acquired the e-tickets. This assumes that the underlying e-tickets are transferable.

Get-in

When a user is about to begin a new journey (after making a transfer or after the activation of an e-ticket), she holds her mobile phone near to a device located at the entrance control system. Both devices communicate through a proximity communication system. If the user provides a *ready to be used travel token* (signed under both the "Get-in" and "Get-out" key pairs) and a valid *deposit token*, she will be allowed to enter. The transport system, for each trip, generates a unique identifier ID_{trip}. The user receives an *inspection token* that will be required in case of inspection.

The user is required to leave a *deposit token*. In this way, she will be forced to run the "Get-out" process at the end of her journey so as to obtain a new one. If not done, she will lose it and will have to pay for a new one.

1. The app transmits its *deposit token*, namely $\{deposit, \text{Sign}_{S^d}(\mathcal{H}(deposit))\}$.
2. The entrance device verifies the digital signature over $\mathcal{H}(deposit)$ and asks the system server to check it has not been used before. If all these checks are satisfied, the server stores $\mathcal{H}(deposit)$ in a database to record it as already used.
3. The app transmits a *ready to be used travel token* TT_i (accompanied with $\text{Sign}_{S^{in}}(\mathcal{H}(TT_i))$ and $\text{Sign}_{S^{out}}(\mathcal{H}(TT_i))$) to the entrance device.
4. The entrance device verifies both digital signatures and checks that $TT_i \neq \mathtt{TT_{end}}$.

5. The entrance device computes the digest $\mathcal{H}(TT_i)$ and sends it to the system server to verify that it has not been used before. If not used before, the server stores $\mathcal{H}(TT_i)$ together with $CurrentTime$ to record it as already used.
6. The app proves in zero-knowledge to the entrance device that the commitment C_i extracted from TT_i contains an expiration time that is still valid by proving that it falls between $CurrentTime$ and $CurrentTime +$ `TIME_ACTIVE`. This is done by means of the techniques described in Section 2.8.
7. The entrance device extracts $NextTT_i$ from TT_i and asks the server to sign it with the "Get-in" key, so that the app blindly obtains a signature $\text{Sign}_{S^{in}}(\mathcal{H}(TT_{i+1}))$.
8. The entrance device extracts U_i from TT_i and asks the server to sign it with the user-token key, so that the app blindly obtains $U_{token} = \text{Sign}_{S^u}(\mathcal{H}(\text{HMAC}_{K_i}(ID_{masked})))$.
9. The user asks, via the entrance device, the server to compute a partially blind signature on U_{token} with ID_{trip} as agreed information. The resulting signature $U_{trip} = \text{PartialSign}_{S^t}(\mathcal{H}(U_{token}), \mathcal{H}(ID_{trip}))$ is obtained by the app.
10. Finally, the *inspection token* is stored by the app as the tuple $\{U_{token}, ID_{trip}, U_{trip}\}$.

Get-out

When a user has finished a journey and is about to leave the transport system, she has to hold her mobile phone close to the corresponding exit device. As a result of this protocol, the user obtains her next *travel token ready to be used* (she receives a second signature under the "Get-out" key pair) and a new free of charge *deposit token* to be used in her next journey. She is also asked to present her *inspection token* so as to record it as no longer valid.

If this protocol is not performed, the user does not obtain a *deposit token* and will have to pay for a new one so as to use the public transport again in the future.

1. The app hashes and blinds, with a random number t, the next *travel token* TT_{i+1}. Then it sends the result, $\text{Blind}_t(\mathcal{H}(TT_{i+1}), P_{S^{out}})$, to the exit device.
2. The exit device asks the system server to sign the received blinded message under the "Get-out" key pair so that the app obtains $\text{Sign}_{S^{out}}(\mathcal{H}(TT_{i+1}))$.
3. The app obtains a free *deposit token* as follows:
   (a) The app sends the current *inspection token* $\{U_{token}, ID_{trip}, U_{trip}\}$ to the exit device together with $\text{HMAC}_{K_i}(ID_{masked})$.
   (b) The exit device checks ID_{trip} corresponds to the current trip of the transport system.
   (c) The exit device, in collaboration with the system server, verifies that neither U_{token} nor U_{trip} have been received before. It also verifies that U_{token} is a valid signature on $\mathcal{H}(\text{HMAC}_{K_i}(ID_{masked}))$. It finally verifies that U_{trip} is a partial signature $\text{PartialSign}_{S^t}(\mathcal{H}(U_{token}), \mathcal{H}(ID_{trip}))$.
   (d) By running the procedure in Section 4.4.4, the app obtains a new deposit token free of charge, $\{deposit, \text{Sign}_{S^d}(\mathcal{H}(deposit))\}$.
   (e) The system server stores $\{U_{token}, U_{trip}\}$ so as to record them as no longer valid.

Random inspection

During an inspection, an inspector determines if a user has a valid *inspection token* for the current public transport trip.

1. The app sends $\{ID_{user}, ID_{trip}, U_{token}, U_{trip}, K_u, K\}$ to the inspector via a short-range communication system.
2. The inspector:
   (a) Checks that the data in ID_{user} correspond to the person in front of him (it may be a picture or her passport number).
   (b) Checks that ID_{trip} corresponds to the current trip.
   (c) Checks that neither U_{token} nor U_{trip} have been provided in a previous inspection and have not been marked as no longer valid.
   (d) Computes $ID_{masked} = \text{HMAC}_{K_u}(ID_{user})$, and checks that U_{token} is a valid signature $\text{Sign}_{S^u}(\mathcal{H}(\text{HMAC}_K(ID_{masked})))$.
   (e) Checks that U_{trip} is a valid partial signature $\text{PartialSign}_{S^t}(\mathcal{H}(U_{token}), \mathcal{H}(ID_{trip}))$.
3. The inspector sends both U_{token} and U_{trip} to the server so as to record them as no long valid.

If any of these checks fail, the user will be fined according to the established regulations. After being inspected, the user obtains a new *inspection token* so as to avoid her next execution of the "Get-out" process being linked to the inspection. This is done as follows:

1. The app generates a random key K', a random blinding factor u', and computes $U' = \text{Blind}_{u'}(\mathcal{H}(\text{HMAC}_{K'}(ID_{masked}))), P_{S^u})$ which is sent, via the inspector's device, to the server.
2. The server blindly signs U' and returns, via the inspector's device, the result to the app so that it can obtain $U'_{token} = \text{Sign}_{S^u}(\mathcal{H}(\text{HMAC}_{K'}(ID_{masked})))$.
3. Next, the app asks, via the inspector's device, the server to compute a partially blind signature over U'_{token} with ID_{trip} as agreed information. The resulting signature $U'_{trip} = \text{PartialSign}_{S^t}(U'_{token}, ID_{trip})$ is obtained by the app.
4. Finally, the app stores the new *inspection token* as the tuple $\{U'_{token}, ID_{trip}, U'_{trip}\}$.

4.5 Privacy objectives and privacy analysis

4.5.1 Privacy objectives

The privacy goals to be achieved by our proposal are listed below:

1. While traveling, the interactions with the system (except for those arising from an inspection carried out by an inspector) performed by a user are anonymous and unlinkable.
2. While traveling, a user is only required to identify herself if requested by an inspector. In such a case, the only information obtained by the system is a *boolean*

indicating whether the identified user is allowed to make the current journey.

Privacy objective 1 implies that the moments a user enters and leaves a transport system cannot be related. Also, the system cannot link the different journeys made by a user with one or several e-tickets.

Privacy objective 2 determines that, as a result of an inspection, the system can only obtain information inherent in the inspection itself. That is, it can associate a user with a specific time and place. Any additional information, like the starting and end stations, cannot be deduced.

4.5.2 Privacy analysis

We next present a set of lemmas which are the basis for proving Theorem 4.5.5.

Lemma 4.5.1. *During an execution of the "Travel token chain generation" procedure, the server obtains no information allowing it to identify the generated chain.*

Proof. A *travel token* chain is generated through a cut-and-choose protocol in which the app first generates $\mathtt{N}$ blinded chains. After that, the server asks the app to reveal all the parameters used in the generation of $\mathtt{N}-1$ of them. Finally, the first *travel token* of the remaining chain is blindly signed by the server. We next justify that the server obtains no information, allowing it to identify the remaining chain.

All the random parameters generated during the creation of the $\mathtt{N}$ chains are chosen independently among them so that revealing the parameters of $\mathtt{N}-1$ chains does not provide any information about the parameters used to create the remaining one.

The ID_{user} object is masked with a different random key before its inclusion in each chain (set $\{ID_{masked_k}\}_{0\leq k<\mathtt{N}}$). These masked values are revealed for $\mathtt{N}-1$ of the chains, but their knowledge does not provide any information about the remaining one, since the random keys are kept secret.

The $\mathtt{N}$ chains share the same expiration time E which is revealed during the cut-and-choose checks. Nevertheless, E is included in the travel tokens TT_i through commitments $C_i = \mathrm{Commit}_{c_i}(E)$. In all future uses of these tokens, the non-expiration of a *travel token* is proven by means of zero-knowledge proofs so that the exact value for E is never revealed. Hence, the remaining chain cannot be identified from E.

Finally, the only access the server has to the remaining chain is produced when it is asked to blindly sign its first *travel token*. A blind signature protocol guarantees that the signing party obtains no information about the signed data. Hence, the server obtains no information about the signed token.

Lemma 4.5.2. *The* travel tokens *of a chain cannot be linked among them by the server.*

Proof. A *travel token* TT_i is a tuple composed of three parameters, namely U_i, C_i, and $NextTT_i$.

Both U_i and $NextTT_i$ are blinded values generated from random blinding factors so that they cannot be related to the actual values blinded inside them or to the resulting signatures. Hence, they provide no information at all as long as the blinding factors are kept secret by the app.

The expiration time, E, committed in C_i is shared among all the *travel tokens* of a chain. However, each C_i is generated from a different and random c_i so that linking *travel tokens* from these commitments is computationally unfeasible. The non-expiration of a *travel token* is proven through zero-knowledge proofs so that the actual value E is not revealed.

Lemma 4.5.3. *When an* inspection token *is generated, the server only obtains information about its ID_{trip} component.*

Proof. An inspection token is a tuple $\{U_{token}, ID_{trip}, U_{trip}\}$. The U_{token} component is obtained by the app as a blind signature computed by the system server. Hence, the server obtains no information about it.

Regarding U_{trip}, it is obtained by the app as a partially blind signature over U_{token} with ID_{trip} as agreed information. Hence, the server only learns ID_{trip}.

Lemma 4.5.4. *After a "Random inspection", the "Get-in" and "Get-out" procedures of the inspected journey cannot be determined.*

Proof. During an inspection, the app sends the tuple $\{ID_{user}, U_{token}, ID_{trip}, U_{trip}, K_u, K\}$ to the inspector.

The link between an *inspection token* and the *travel token* TT_i provided for its generation is given by U_{token} being a signature over the value blinded in the U_i component of TT_i. Since U_i is a blinded value, it cannot be related to the (unblinded) signature obtained from it. Hence, an *inspection token* provided during an inspection cannot be related to the execution of the "Get-in" process in which it was generated.

At the end of an inspection, a new *inspection token* is generated by the user. When it is provided during an execution of the "Get-out" procedure, it will not be linkable to the current inspection.

Theorem 4.5.5. *The proposed system meets the design objectives 1 and 2 (user's mobility profiles cannot be created).*

Proof. When a user activates an e-ticket of the underlying system, a chain of *travel tokens* is created. From Lemma 4.5.1, the server obtains no information allowing it to identify that chain; hence, the "e-Ticket activation" operation remains anonymous and unlinkable.

After that, each time the user runs a "Get-in" procedure, she provides a *travel token* of that chain. From Lemma 4.5.2, the tokens of a chain cannot be related among them so that the executions of the "Get-in" process using tokens of the same chain cannot be related among them.

When running a "Get-in" process, the user also obtains an *inspection token*, from which, according to Lemma 4.5.3, the server obtains no information but its ID_{trip} component. Hence, the server obtains no information at all about the user from the "Get-in" process.

When running a "Get-out" process, the user provides her *inspection token* so as to mark it as no longer valid. Since the keys used to mask user's identity in that token are not revealed, the server cannot link that token to the identity of the user. From Lemma 4.5.3, the provided *inspection token* cannot be linked to any personal data of the user.

When a random inspection takes place, the user is asked to provide her *inspection token* and all the data allowing it to be linked to her identity. The inspector obtains user's identity but, according to Lemma 4.5.4, her *inspection token* cannot be linked to the executions of the "Get-in" and "Get-out" procedures of that journey. In this way, the only information obtained is the presence of the inspected user at the place where the inspection has taken place.

4.6 Experimental results

The actors considered in our experiments are: the *server*, the *entrance devices*, the *exit devices*, and the app. To simplify, we have considered the *entrance* and *exit devices* to be part of the *system server*.

Prototypes have been implemented in `Java`. The *app* has been developed specifically for the `Android` operating system. The `java.math.BigInteger` library has been used to implement cryptographic operations involving big integers.

The feasibility of the system has been analyzed from the two most critical use cases. Namely, "e-Ticket activation" (Section 4.4.4) which performs a cut-and-choose check involving strong computations, and the "Get-in" process (Section 4.4.4) which should respond with a minimum delay. Both procedures are executed between the app and the *system server*.

Our experiments employ 2048 bit cryptography for RSA blind signatures and for all the operations involving hard-to-solve instances of the discrete logarithm problem. Hash digests are computed by means of the SHA-256 function.

The cut-and-choose check run during "e-Ticket activation" has been tuned to $\mathtt{N} = 100$ so that the probability of cheating is reduced to 1% (generation of a fraudulent chain involves, on average, the payment of 99 fines). The "Travel token chain generation" procedure has been executed with parameter $\mathtt{T} = 3$ (maximum amount of journeys which can be made with a single e-ticket). All the implemented processes require intensive CPU usage with negligible memory requirements.

Table 4.3 shows the average running times for the app. Two variants of the "e-Ticket activation" procedure have been tested. A *real-time* variant in which all the computations are performed during the protocol execution, and a *pre-computed* variant in which values for ID_{masked}, U_i, and $NextTT_{\mathtt{T}-1}$ have been computed in advance. Both variants have

Table 4.3: Mobile application running times (in milliseconds).

Mobile phone			**e-Ticket activation**		**Get-in**
BQ Model	Cores	GHz	Real-time	Pre-computed	Serial
Aquaris U Lite	4	1.4	21528	8952	880
Aquaris U	8	1.4	16414	4467	877

been implemented taking advantage of parallel computing. We can observe that pre-computations provide a 60%-75% time reduction. The "Get-in" protocol of the app has been implemented in serial mode (it cannot benefit from either pre-computations or parallel processing). Running the "Get-in" procedure took less than one second in the two devices on which it was tested.

Table 4.4: System server running times (in milliseconds).

System server				**e-Ticket activation**		**Get-in**	
Processor	Cores	Threads	GHz	Serial	Parallel	Serial	Parallel
Intel i5-4310	2	4	3.00	17163	7719	320	148
AMD Athlon	4	4	2.80	25867	7497	487	126
Intel i7-6700	4	8	3.40	11812	2814	219	56
Intel i7-8700	6	12	3.20	10103	1615	184	30

Table 4.4 shows the average running times of the server. Both the "e-Ticket activation" and the "Get-in" procedure have been executed in serial and in parallel. In the parallel version, multiple instances of the "e-Ticket activation" procedure have been executed concurrently.

The experiments show that the app can be deployed on current mobile phones, since tickets can be activated in around 4.5 seconds, whereas a "Get-in" procedure can be completed in less than one second. The computer load at the server part must be balanced among several computers so as to manage all the concurrent requests received in a large city. A professional 36-Core (72-Thread) computer could manage three "e-Ticket activation" requests per second.

Chapter 5

Efficient pay-by-phone parking system

This chapter presents a system summarized in the scientific paper *"An efficient privacy-preserving pay-by-phone system for regulated parking areas"* which was published in the International Journal of Information Security (2020) [BS20b]. It aims to reduce the computational cost of proposal [BS19] described in Chapter 3.

5.1 Introduction

Traditionally, when parking in regulated areas, drivers had to purchase a ticket from a machine and display it on the dashboard of the car.

Smartphones can currently run applications that allow these payments to be made much easier [20c; 20d; 20e; 20f; 20j; 20h; 20a; 20i]. Upon parking, the driver introduces the license plate number of the car, the expected parking duration, and the parking area. A digital payment is then performed using a credit card or deducted from a pre-paid balance. These applications provide several advantages: they avoid the use of paper and eliminate the need to move to a pay station or carry coins. They also allow the parking time to be extended without returning to the car, while some of them even refund the money corresponding to unused time if the car is removed earlier than expected.

In order to avoid the automatic tracking of vehicles, the payment status of a car should not be checkable unless a parking officer is located close to the parked car.

So as to obtain the desired privacy, it is necessary to use and combine cryptographic tools and protocols. These cryptographic tools and protocols may require elevated computing resources. When the time spent by the protocols is too high for the solution to be deployed, the proposed system has to be optimized or redesigned.

5.2 Related work

Several pay-by-phone parking systems have been deployed in many cities [20c; 20d; 20e; 20f; 20j; 20h; 20a; 20i] [1]. They work properly and make payments easy for drivers. Nevertheless, they collect very complete and linkable (from car plate numbers) information about all the parking transactions. Hence, parking profiles can easily be built from their data if stolen or leaked by an internal attacker.

There exist academic proposals addressing the privacy of drivers by means of cryptographic protocols [Pér15; GMS17; BS19]. All these proposals share a similar system model and user experience. They all implement a paradigm in which drivers acquire pre-paid credit in the form of anonymous e-coins. Parking transactions are then paid by e-coins. Drivers are required to incorporate an on-board RFID device into cars which is queried by parking officers when an inspection takes place.

In [GMS17] the mobile application of drivers performs successive micro-payments for short-time intervals while their cars are parked. These payments cannot be linked among them, and can only be related to the car as a result of an inspection carried out by an officer. Its main disadvantage comes from the fact that if some of the micro-payments could not be made due to a lack of coverage or low battery, the driver may be fined. Hence, that system requires drivers' devices to be permanently connected. The previous issue is addressed in [BS19]. Upon parking, the driver pays in advance for a constant amount of time intervals. So as to mask the expected duration of parking, e-coins can be *valued* or *no-valued*. Valued e-coins are used to pay for those intervals in which the driver expects to be parked, while the remaining ones are paid with no-valued e-coins. As a result of an inspection, the system just obtains a boolean indicating whether a payment with a valued e-coin was made for the current time interval.

In our opinion, the proposal [BS19] described in Chapter 3 is the most complete privacy-preserving pay-by-phone parking system. However, the security of some of its procedures holds on cut-and-choose checks tuned with a security parameter `N`. They reduce the probability of cheating to $1/\mathtt{N}$ at the expense of increasing the computational cost linearly with `N`. Hence, only moderately small probabilities of cheating can be attained at a reasonable cost.

The proposal [BS19] uses the cut-and-choose technique at two moments: during the generation of no-valued e-coins and at the beginning of a parking transaction. No-valued e-coins can be generated in a background batch procedure. However, ticket generation must be done in real time so that the time spent by this process is as short as possible.

5.3 Cut-and-choose technique

A blind signature protocol (Section 2.5) ensures that the signer does not learn anything about the signed message. There are, however, some situations in which the signer should

[1] A detailed review of related work on privacy-preserving pay-by-phone parking systems can be found in Section 3.2.

check that the message he is going to sign satisfies some property. Such checking may be done by means of the cut-and-choose technique [Sch96].

Next, we show a simple example in which the signer Bob wants to verify that the message he is going to blindly sign really contains an odd random number:

1. Alice generates N messages which are blinded, each with a different blinding factor. Then she sends the N blinded documents to Bob.
2. Bob chooses $N-1$ documents at random and asks Alice to send the blinding factor for each of the selected documents.
3. Alice sends the appropriate blinding factors.
4. Bob unblinds the chosen $N-1$ documents and checks that each of them contains an odd number.
5. Bob blindly signs the remaining document and sends the result to Alice.
6. Alice unblinds the received signature.

The main drawback of this technique lies in the fact that it requires the generation and verification of N-1 additional messages. Their generation and verification can be computationally high especially if such checks involve modular computations over big integers. The probability of fraud is $(1/N)$, whereas the computational cost grows linearly with N.

5.4 System and adversary models

Like in [GMS17] and [BS19] the current proposal divides the time into short-duration intervals (*e.g.* 5 or 10 minutes) and drivers (micro)pay for those slots during which their cars are parked. Each time slot is represented as a unique integer indicating, for instance, the amount of intervals elapsed since the beginning of year 2000 (or any other chosen reference time). In this way, each ticket is issued linked to the corresponding slot identifier. Just after parking, the driver, by means of the app, will obtain tickets for those slots in which her car is expected to be parked. Tickets are acquired anonymously using pre-paid e-coins. Unused tickets can be revoked in advance and refunded with the amount paid.

This section describes the system and adversary models. After that, it enumerates the requirements to be satisfied by a privacy-preserving pay-by-phone parking system.

5.4.1 System model

The actors that compose the system model (Figure 5.1) are:

1. **Mobile application** or **app.** This is installed and run on drivers' mobile devices. The app interacts with the system server during the purchase of pre-paid e-coins, and during the issuance and revocation of tickets. In case of inspection it is requested to provide a valid ticket.

2. **System server.** This is an on-line platform accessed by apps and officers through the Internet. It interacts with an app when acquiring e-coins and while issuing or revoking tickets. During an inspection, it is requested by officers to ask for a valid ticket from the corresponding app.
3. **On-board device.** Each car must incorporate an on-board RFID device. It is accessed by officers during a parking inspection.
4. **Officer.** He patrols the city checking the payment status of cars. An officer queries the on-board device of the inspected car and then asks the server to determine whether or not a ticket exists for that car.
5. **Timestamp authority** or **TSA.** This issues timestamps when requested by the server. Its role is to avoid disputes over double-spent e-coins or revoked tickets.

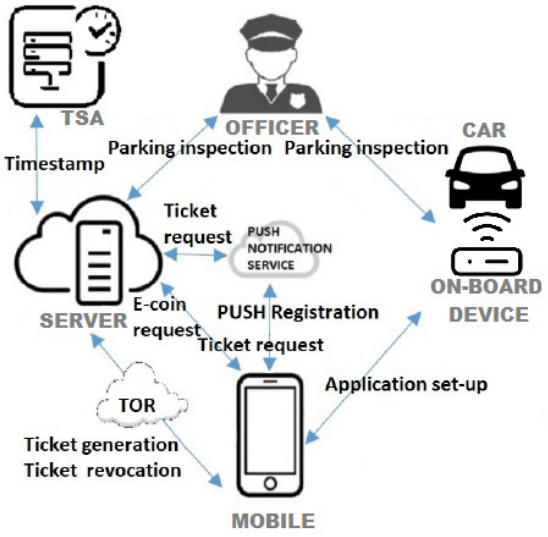

Figure 5.1: System model

Since tickets for parking are issued for specific time intervals, all the actors' clocks must be synchronized.

A *push notification service* [BM16; RK13] allows data to be sent from servers to applications. In our system, the app requests an `ID_PUSH` identifier which is sent to the server together with the car plate number. In this way, the server will be able to send messages to the corresponding app.

5.4.2 Adversary model

Attacks against the cryptographic primitives are assumed to be unfeasible by setting appropriate key sizes.

Regarding the capacity of an attacker aiming to attack the privacy of drivers:

- **Assumption 1.** An adversary cannot corrupt the app or the on-board device.
- **Assumption 2.** The server and the officers follow the defined protocols but may collaborate with an adversary by providing the data to which they have access.

The first assumption is the basis for any secure protocol to make sense. Secure protocols cannot protect privacy when run on devices infected with malware which may be transmitting private data directly to the attackers. Integrity of the on-board device can be achieved by means of tamper-proof hardware.

Regarding the second assumption, any deviation from the correct execution of the protocols made by the system will be immediately detected by the app. After acquiring e-coins or generating tickets, the app receives signed data whose correctness is immediately validated. When an inspection takes place, it also receives data signed by driver's on-board unit which can easily be checked. Attacks based on interrupting the system procedures execution, or the issuance of unfair parking fines by parking officers, would lead drivers to complain to the city council asking for an alternative service provider. Hence, assuming that in case of corruption they take honest but curious behavior is realistic.

From the service provider side, drivers are untrusted *actors* who might try to obtain parking time for free. Both the app and the on-board device may deviate from protocol specifications. Their motivation is to obtain parking time for free or avoid the payment of a fine.

The communications among all the actors should be carried out over a secure transport protocol, such as TLS [Res18], to avoid external attacks like server spoofing or session hijacking, and guarantee the confidentiality of all the transmitted data.

5.4.3 Design objectives

This section enumerates and describes the design objectives. Objectives 1 and 2 are equivalent to those in [GMS17] while objective 3 was implicitly assumed in [BS19]. Objective 4 was first proposed in [BS19]. They are:

1. The parking status of a car can only be determined as a result of an inspection by an officer located close to it. The only information obtained by the system during an inspection is a *boolean* indicating whether a payment for the present time and for the inspected car was performed.
2. A driver can prove to a third party that a payment for a given time interval was made. In that case, no information about other time intervals is revealed.
3. A ticket can be revoked in advance without revealing the car for which it was issued.
4. The app only needs an Internet connection at the beginning of a parking transaction or when revoking unused tickets.

Objective 1 guarantees that the payment status of a car cannot be tracked automatically (an officer has to be in situ). It also determines that the information collected by the system provides an advantage to the creation of parking profiles. The information that an attacker can obtain is exactly the same as they would obtain by roaming parking places and collecting information about parked cars.

Objective 2 protects drivers against unfair fines while also guaranteeing that no additional information is revealed in such a situation. It also allows tickets to be used as

parking receipts.

Objective 3 allows recovery of the money from unused parking time without any privacy concern.

Objective 4 states that drivers who paid to park their cars will not be fined due to their mobile phones being out of coverage during a parking inspection.

5.5 System proposal

5.5.1 System overview

A driver installs an app on her mobile device which manages an electronic wallet that is pre-loaded with e-coins that can be *valued* or *no-valued*. Valued e-coins are acquired by running the "E-coin request" procedure. The price of a valued e-coin corresponds to the price for parking during a time slot. They are purchased in batch and paid via some online payment method like credit card. No-valued e-coins are generated by the app itself at no cost. When a driver spends an e-coin, the system is unable to determine whether the received e-coin is valued or no-valued.

Upon parking, a driver obtains a ticket for each of the (consecutive) slots that compose the expected parking time by running the "Ticket generation" procedure. Each ticket is paid anonymously via a valued e-coin and stored in the app.

So as to hide the expected parking duration, the app always requests the same amount of tickets. Those tickets for which the driver expects to be parked are paid using valued e-coins while the remaining dummy ones are paid with no-valued ones. The app does not obtain tickets from payments performed with no-valued e-coins. Therefore, the app only obtains a ticket for those slots for which it has paid (Figure 5.2).

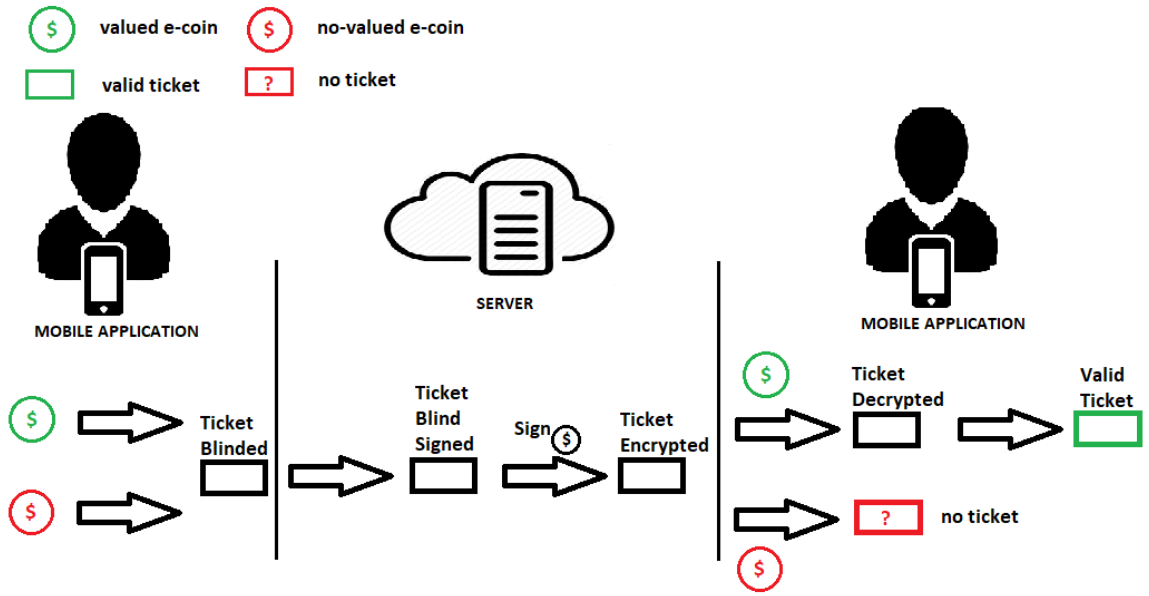

Figure 5.2: Abstract graphic

When an inspection takes place, the officer, by running the "Parking inspection" procedure, first queries the on-board device of the inspected car and then asks the server to determine the payment status of that car. The server then runs the "Ticket request" procedure so as to ask the app for the corresponding ticket.

If the car is removed earlier than expected, unused tickets can be revoked through the "Ticket revocation" procedure. A valued e-coin is provided in exchange for each revoked ticket.

5.5.2 System description

The procedures that compose our pay-by-phone parking system are detailed below:

System parameters generation

Prior to its deployment, some parameters have to be created and set.

1. The system server generates the following cryptographic keys:
   - An "E-coin" public/private RSA key pair: $(N_C, e_C)/d_C$.
   - A "Ticket" Partially Blind Signature public/private key pair: P_S/V_S (Section 2.6).
2. The system server chooses an elliptic curve E with a large prime cardinality q, and publishes a generator P of its group of points.
3. The *timestamp server* generates:
   - A "TSA" digital signature public/private key pair: P_{TSA}/V_{TSA}.

All the public keys must be certified by a certificate authority and be made publicly available. The private keys must be kept secret by the corresponding server.

Remark. The "E-coin" key pair will be used to issue e-coins. The "Ticket" one will be employed for the issuance of tickets. Timestamps are to avoid disputes about double-spent e-coins or revoked tickets.

Application set-up

Each driver must install and configure the app on her mobile phone. During this setting process, the app connects to the on-board device and sends the current time and the car license plate number.

The on-board device then sets its internal clock, stores the plate number, and generates a digital signature key pair P_D/V_D. The private key V_D is stored internally while the public key P_D is transmitted to the app, which stores and forwards it to the server.

Remark. This key pair will be used by the on-board device to sign messages which prove that an officer is located close to the car during an inspection.

E-coin request

E-coins are data-signature tuples signed under the "E-coin" RSA key-pair. Let d_C be the private key, while (N_C, e_C) is the public one.

1. A valued e-coin is generated as follows:
   (a) The app performs a credit card payment for the price of an e-coin[2].
   (b) The app generates a random value $v \in [0, q-1]$.
   (c) The app computes $Q = vP$ and $C = \text{Embed}(Q)$.
   (d) The app chooses $r \in \mathbb{Z}_{N_C}$ at random and computes $C' = \text{Blind}_r(C)$. Then, C' is sent to the server.
   (e) The server signs C' under private key d_C and sends the resulting signature S' to the app.
   (f) The app computes $S = \text{UnBlind}_r(S')$ so that $\{C, S\}$ is a data-signature tuple.
   (g) The app stores the e-coin tuple $\{C, S, v\}$.
2. A no-valued e-coin is generated as follows:
   (a) The app takes $S \in \mathbb{Z}_{N_C}$ at random.
   (b) The app computes $C = S^{e_C} \pmod{N_C}$.
   (c) The app computes $Q = \text{Embed}^{-1}(C)$. If the procedure fails (Section 2.7.2), the app goes back to step 2a.
   (d) The app stores the tuple $\{C, S\}$. Note that $v = \log_P(Q)$ is unknown to the app.

Remark. The difference between a valued and a no-valued e-coin lies in whether or not v is known.

Ticket generation

A ticket to park a car with plate number *License* during a given *Slot* is generated as follows:

1. The app takes an unspent e-coin (valued or no-valued) and sends its public part $\{C, S\}$ to the server.
2. The server checks that $\{C, S\}$ is a valid message-signature tuple under the "E-coin" public key.
3. The server verifies that the received e-coin has not been spent before. In such a case, its previous timestamp would be returned as proof. Otherwise, it asks the TSA to issue a timestamp on it, and stores its S component together with its timestamp.
4. The server obtains the elliptic curve point $Q = \text{Embed}^{-1}(C)$.
5. The app generates a random key K and computes $L = \text{HMAC}_K(License)$. Then, the app and the server engage in the computation of a partially blind signature over L with *Slot* as agreed information. The server encrypts the data sent during the last round using ECIES under public key Q.

[2]Valued e-coins will usually be acquired in batch so that a single payment for the overall amount will be performed.

6. If the spent e-coin was valued, the app decrypts, using v, the data received from the last round so that it can obtain a partial signature $T = \text{PartialSign}_S(L, Slot)$. Then, the app stores the ticket which consists of tuple $\{L, Slot, K, T\}$.
7. Otherwise, if the spent e-coin was no-valued (private key v is unknown), the partially blind signature cannot be obtained and all the data are discarded.

Remark. The server cannot determine whether or not the app has obtained the ticket.

Parking inspection

The officer inspects the parking status of a car as follows:

1. The officer queries the on-board device.
2. The on-board device obtains the current *Slot* from its internal clock. Then, it computes the digital signature $\text{Sign}_D(\mathcal{H}(Slot||License))$. The values for *Slot*, *License*, and their signature are sent to the officer.
3. The officer checks the received values for *Slot* and *License*. If they are correct, he forwards the tuple $\{Slot, License, \text{Sign}_D(\mathcal{H}(Slot||License))\}$ to the server.
4. The server obtains the public key, P_D, and the push information associated with *License*. Then, it verifies the signature over *Slot* and *License* with public key P_D. If it is correct, the server sends a push message to the app.
5. The server expects to receive a valid ticket from the app by means of the "Ticket request" procedure.
6. If such a ticket is not received the driver will be fined and notified by telematic means.

Remark. Possession of $\text{Sign}_D(\mathcal{H}(Slot||License))$ is proof that the officer was physically close to the car during the inspection.

Ticket request

When the app is notified about an inspection for a specific *Slot*, it runs the following procedure:

1. The app asks the server to send the signature computed by the on-board device over *Slot* and *License*.
2. The app checks the signature on *Slot* and *License* with the on-board public key P_D. If the verification is correct, the app sends the ticket tuple $\{L, Slot, K, T\}$ to the server.
3. The server checks that L equals $\text{HMAC}_K(License)$ and verifies the partial signature $T = \text{PartialSign}_S(L, Slot)$ under public key P_S.
4. The server also verifies that T has not been previously revoked. In such a case, the server would return its timestamp generated during the "Ticket revocation" procedure as proof.

Remark. If the app cannot provide a valid ticket, the driver will be fined (step 6 of the "Parking inspection" procedure).

Ticket revocation

The driver can revoke unused tickets and obtain valued e-coins in exchange. The app connects to the server and runs the following procedure for each ticket to be revoked:

1. The app sends the tuple $\{L, Slot, T\}$ to the server.
2. The server verifies that:
   (a) T is a partially blind signature $\text{PartialSign}_S(L, Slot)$ computed under public key P_S.
   (b) $Slot$ is a future time slot.
   (c) T has not been revoked previously. If it has, its previous timestamp is returned as proof.
3. If the ticket can be revoked, the server requests the TSA to timestamp its T component and stores the tuple $\{T, \text{Timestamp}_{TSA}(T)\}$.
4. Finally, the app and the server engage in an execution of the "E-coin request" procedure so that the app obtains a valued e-coin.

Remark. The car plate number is not revealed during the ticket revocation process.

5.6 Privacy and security analysis

This section analyzes the security provided by the proposal under the assumed adversary model (Section 5.4.2) by proving that it fulfills the design objectives (Section 5.4.3).

5.6.1 Privacy analysis

When a driver parks, her app requests a constant amount of tickets through the "Ticket generation" procedure. At this moment, the server knows that a parking transaction has just begun, but it must be unable to determine: (1) The plate number of the car involved; (2) Which tickets were paid with valued e-coins (so that the expected duration is kept secret).

Tickets are stored secretly in the app. They are only provided in two cases:

- When an officer located close to the car checks its parking status. The driver's app is then asked to provide the ticket corresponding to the current time slot ("Ticket request" procedure). At this moment, the time at which the ticket provided was issued should not be revealed so as to keep the parking starting time secret.
- When complaining about an unjust fine. Like before, the ticket issuance time has to be kept secret.

- When revoking tickets in advance. In this case ("Ticket revocation" procedure), both the issuance time and the car plate number must remain secret.

By further requiring that a ticket only provides information about the time slot it was issued for, and by having tickets digitally signed by the server (they can be proven to be authentic) the previous statements lead to the fulfillment of the design objectives. The following lemmas and theorem address them.

Lemma 5.6.1. *The information collected by the server during the issuance of a ticket ("Ticket generation" procedure) cannot be linked to the car license plate number.*

Proof. When a valued e-coin is issued ("E-coin request" procedure), the server blindly signs it, so that it obtains no information about it. No-valued e-coins are issued by the app with no interaction with the server, so that the server does not obtain any information about them either. Hence, when a driver spends an e-coin during the issuance of a ticket, that e-coin (valued or not) cannot be linked to the moment at which it was generated or to the driver who is providing it.

When a ticket is issued, its content is generated and stored by the app which asks the server to compute a partially blind signature on it. Hence, the only information the server obtains about an issued ticket is the time slot it was requested for (the agreed information of the partially blind signature).

The "Ticket generation" procedure is executed through an *anonymous channel* which avoids tracking from the underlying communication protocols.

From Assumption 1, the app cannot be corrupted, so that the system will not obtain any information but that received from a proper execution of the protocols.

□

Lemma 5.6.2. *The information collected by the server during the issuance of a ticket ("Ticket generation" procedure) does not allow it to be determined whether or not a ticket was obtained by the app.*

Proof. The app should only obtain a ticket if it has paid with a valued e-coin. When a ticket is issued, the app spends an e-coin by sending the tuple $\{C, S\}$ to the server (step 1 of the "Ticket generation" procedure).

The difference between a valued and a no-valued e-coin lies in whether or not the app has knowledge of the private key $v = \log_P Q$. The server cannot determine this. Therefore, the server cannot infer whether or not the received e-coin was valued, so that it cannot determine if a ticket was obtained by the app.

From Assumption 1, the app will not provide any additional information to the server.

□

Lemma 5.6.3. *The information provided by a driver when presenting a ticket ("Ticket request" procedure) for a given time slot cannot be linked to non-inspected slots.*

Proof. The only information obtained by the server from a presented ticket $\{L, Slot, K, T\}$ is the car plate number and the time slot for which it was issued.

Tickets for time slots not checked by an officer are kept secret by the app (from Assumption 1, it will not provide any information about them) so that they cannot be linked in any manner.

□

Lemma 5.6.4. *The information provided by a driver when revoking a ticket ("Ticket revocation" procedure) cannot be linked to the car plate number.*

Proof. When tickets are revoked in advance, only the tuple $\{L, Slot, T\}$ is sent by the app (key K is kept secret). Hence, the car plate number remains secret, since the relation $L = \text{HMAC}_K(License)$ cannot be checked unless K is known. From Assumption 1, key K will never be provided by the app. □

Theorem 5.6.5. *The protocol fulfills the requirements defined in Section 5.4.3.*

Proof. Lemma 5.6.1 states that the information collected by the server when issuing tickets cannot be linked to the car for which they are requested. Lemma 5.6.2 further states that the server cannot determine the expected duration of the parking transaction either. As a result of a ticket request, the app only provides a ticket if the server sends a message signed by the on-board device of the car. Such a signed message can only be obtained if an officer is close to the car. These signed messages include the current time slot, so that they can only be employed to obtain tickets for the current slot. Tickets for other slots are kept secret by the app. Lemma 5.6.3 states that a ticket presented during an inspection provides information only about the inspected slot. Hence, Requirement 1 is fulfilled.

Since a ticket is digitally signed by the server, a driver can provide it as proof of having paid for a given time slot. Lemma 5.6.3 guarantees that the server only obtains the information about the time slot and plate number on the ticket provided. Hence, Requirement 2 is also met.

Lemma 5.6.4 states that no information, other than the time slot it was issued for, is revealed from a revoked ticket. Hence, Requirement 3 is fulfilled.

Since tickets issued for all the time slots composing the expected parking duration are generated at the beginning of the parking transaction, an Internet connection is only required at that moment. If a parking inspection takes place while the app is out of coverage, the app will be allowed to send the ticket later. An Internet connection is also needed for ticket revocation. Hence, Requirement 4 is also satisfied.

□

5.6.2 Security analysis against malicious drivers

This section contains three lemmas and a theorem which allow it to be concluded that a malicious driver cannot obtain parking time for free.

Lemma 5.6.6. *Valued e-coins cannot be forged.*

Proof. An e-coin consists of a data-signature tuple $\{C, S\}$ signed under the server "E-coin" key-pair.

If not obtained (and paid for) through the "E-coin request" procedure for valued e-coins then an attacker must obtain it by forging or simulating it.

If such a tuple has been forged or simulated by the attacker, its C component, when provided as input to the Embed^{-1} procedure (Section 2.7.2), produces a pseudo-random elliptic curve point Q (or a failure) as a result. An e-coin is valued if and only if its owner knows $v = \log_P(Q)$. Since the elliptic curve discrete logarithm is assumed to be intractable, obtaining v for a random point Q is unfeasible. Hence, valued e-coins cannot be forged.

□

Lemma 5.6.7. *The proposed e-coin system is* secure against double spending*.*

Proof. Spent e-coins are timestamped and stored by the server. Before accepting an e-coin, the server checks it has not been spent before. Timestamps allow a previous use of a double-spent e-coin to be proven. □

Lemma 5.6.8. *Tickets cannot be obtained from no-valued e-coins.*

Proof. During the issuance of a ticket ("Ticket generation" procedure), the app and the server engage in the computation of a partially blind signature. The data sent from the server to the app during the last round of the partially blind signature protocol is encrypted using the ECIES scheme under public key Q obtained from the spent e-coin $\{C, S\}$. The app will only be able to decrypt such data if it knows the secret key $v = \log_P Q$. Since v is only known for valued e-coins, the partially blind signature (and the ticket) will only be obtained if the spent e-coin is valued. □

Theorem 5.6.9. *Drivers cannot obtain parking time for free.*

Proof. Lemmas 5.6.6 and 5.6.7 state that valued e-coins cannot be forged or spent more than once. Next, Lemma 5.6.8 guarantees that tickets for parking are only obtained if paid with a valued e-coin. Hence, we can conclude that tickets are only available to drivers who paid for them. □

5.7 Experimental results

5.7.1 Running time

Our experiments have addressed the computing resources needed by the system server and the app when running the "E-coin request" and the "Ticket generation" procedures (Section 5.5.2). The other procedures have a negligible cost.

System prototypes have been implemented in `Java`. The app has been developed for Android using the Java native library `java.math.BigInteger` when required by cryptographic operations involving big integers.

Our implementation employs 2048 bit cryptography for RSA signatures and for all the cryptographic operations involving hard-to-solve instances of the discrete logarithm problem. The key size for elliptic curve cryptography has been set to 224 bits. Hash digests have been computed using the SHA-256 function.

On the app side, the average time needed to generate 12 valued e-coins, 12 no-valued e-coins, and 12 tickets has been measured. On the system server side, we have measured the average time taken to generate 12 valued e-coins and 12 tickets (the server is not involved in the generation of no-valued e-coins). The ticket generation process is exactly the same regardless of the use of valued or no-valued e-coins. Our experiments have involved both serial and parallel executions of all the tested procedures.

Table 5.1: Mobile application running times (in milliseconds)

Mobile phone BQ Aquaris *spec.*			**12 e-coin (valued)**		**12 e-coin (no-valued)**		**12 tickets (valued)**	
Model	Cores	GHz	Serial	Parallel	Serial	Parallel	Serial	Parallel
U Lite	4	1.4	4075	1675	1759	779	8238	3411
U	8	1.4	3766	1206	1804	633	6986	2364

Table 5.2: System server running times (in milliseconds)

System server				**12 e-coin (valued)**		**12 ticket**	
Processor	Cores	Threads	GHz	Serial	Parallel	Serial	Parallel
Intel i5-4310	2	4	3.00	142	72	1402	739
AMD Athlon	4	4	2.80	231	60	2295	635
Intel i7-8700	6	12	3.20	88	16	859	148
AMD Ryzen 7	8	16	4.20	84	10	807	99

Tables 5.1 and 5.2 show the observed running times. Table 5.1 summarizes the experiments run on the mobile application side over two BQ Aquaris mobile phones. Table 5.2 shows the results on the server side obtained over diverse computers with different processors. The first obvious observation is that better running times are obtained from parallel executions. Memory requirements are negligible in all cases.

In our experiments, a mobile phone can generate all the valued e-coins required to issue parking tickets for a two-hour parking transaction (12 slots assuming 10 minutes per slot) in 1206ms (parallel execution on the BQ Aquaris U model). The app can obtain the 12 tickets of such a transaction in 2364ms.

On the system server side, the best running times were obtained on an AMD Ryzen 7 processor with 8-Cores (16-Threads) at 4.20GHz in parallel mode. These results could

be extrapolated to a professional server with 64-Cores (128-Threads) at 4.20GHz. Since such a server would be around 8 times faster, it would be able to issue 9600 valued e-coins per second

$$\left(\frac{12ecoin}{10ms} \cdot \frac{1000ms}{1s} \cdot 8 = 9600\frac{ecoin}{s}\right)$$

or 969 tickets per second

$$\left(\frac{12ticket}{99ms} \cdot \frac{1000ms}{1s} \cdot 8 \approx 969\frac{ticket}{s}\right).$$

Next, we compare our results with those of [BS19] in which a proposal following the same paradigm was presented. The running times in [BS19] were obtained after setting the cut-and-choose security parameter to $N = 100$ (cheating probability is 1%). Let us recall that the running times in [BS19] grow linearly with N.

We first compare the mobile application running times needed to generate 12 e-coins and 12 tickets over the BQ Aquaris U model. Although the new proposal is slower than [BS19] when generating valued e-coins (1206ms against 420ms), it is much faster in the case of no-valued ones (633ms against 26976ms). The time needed for the issuance of 12 tickets has been improved from 13968ms in [BS19] to 2364ms.

Figure 5.3 contains a comparative chart of the mobile application running times between the current proposal and [BS19].

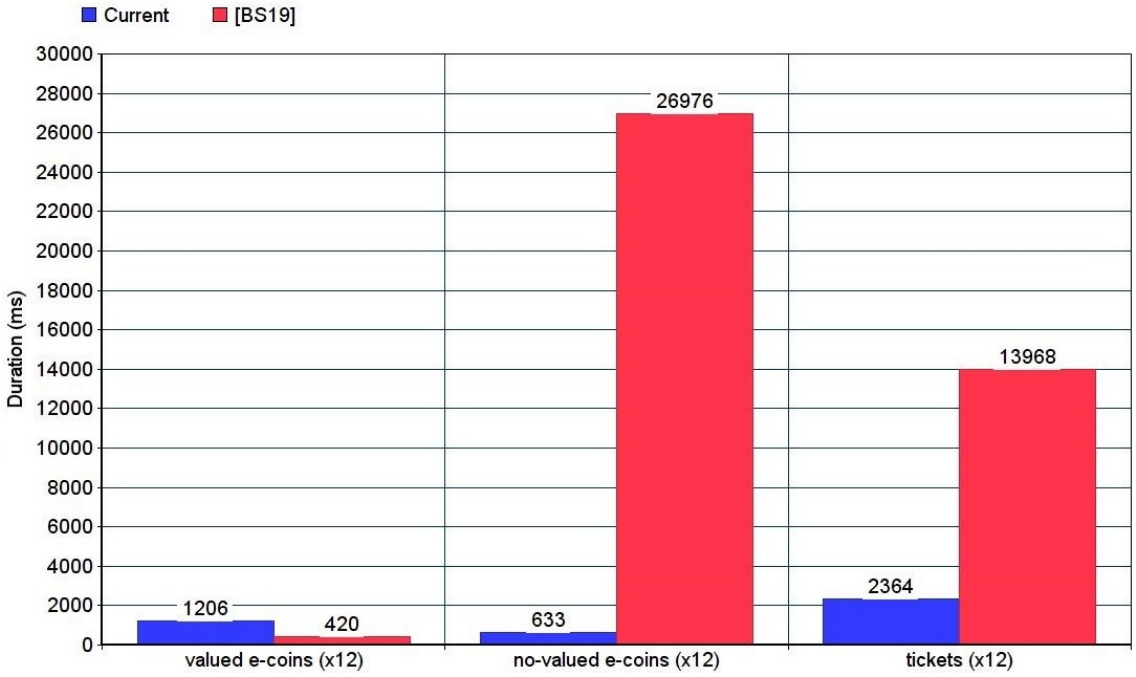

Figure 5.3: Mobile application running times comparison

On the server side, our new proposal is approximately 8.5 times faster than [BS19] (60ms against 504ms on an AMD Athlon server) when issuing 12 valued e-coins, and approximately 80 times faster than [BS19] (635ms against 51192ms) when issuing 12

tickets. In addition, in our new proposal, the server does not participate in the generation of no-valued e-coins. That procedure is rather hard in [BS19] in which the server time required to generate a no-valued e-coin is similar to that required to issue a ticket.

Figure 5.4 contains a comparative chart of the system server running times between the current proposal and [BS19].

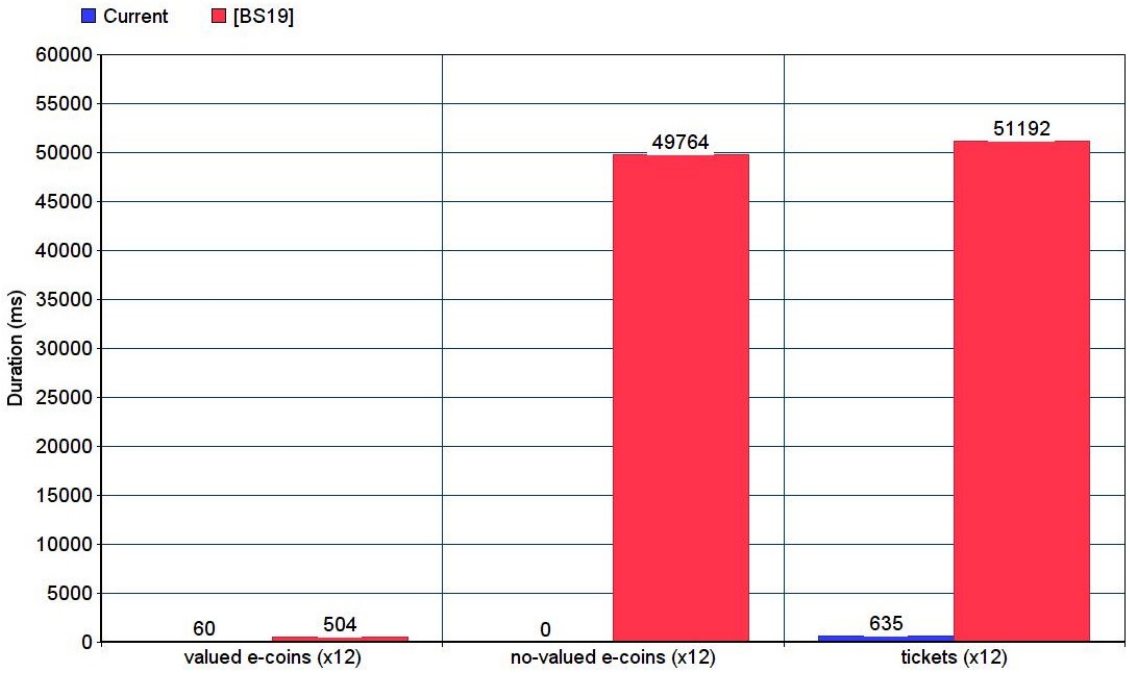

Figure 5.4: System server running times comparison

The obtained running times allow us to conclude that the proposed system could be deployed in a real environment.

5.7.2 Storage requirements

So as to avoid double-spending frauds, the system server has to store a timestamped copy of each spent e-coin (step 3 of the "Ticket generation" procedure). Storage requirements can be reduced by just storing a hash digest of each e-coin together with its timestamp. By using 256 bit hash digests and 2048 bit RSA signatures for timestamps, storing an e-coin takes 288 bytes of space. Let us assume 512 bytes per record to allow the inclusion of some additional metadata.

Assuming 10-minute time slots, each parking bay in a city could generate up to 6 spent valued e-coins per hour or, equivalently, 144 spent valued e-coins per day.

In a large city like Barcelona, with 110,529 on-street parking bays (in 2018), storing the valued e-coins spent during a day would take around 8 GB. Those spent during a whole year would take less than 3 TB.

Let us recall that during a parking transaction some no-valued e-coins are spent for masking purposes. Both types of e-coin are indistinguishable and they have to be stored

in the same way. We can estimate the storage requirements arising from no-valued e-coins by assuming the maximum parking time is 2 hours. If the average parking time was 1 hour, each parking transaction would involve the same amount of valued and no valued e-coins. In this way, storage of no-valued e-coins can be estimated to require around 3 TB.

Hence, 6 TB would suffice to store all the e-coins spent in Barcelona during one year.

Revoked tickets are also stored by the server (step 3 of the "Ticket revocation" procedure). A revoked ticket only needs to be stored until its time slot expires, which should not take longer than the maximum parking time. Assuming the maximum parking time is 2 hours, and each record takes 512 bytes, the storage requirements for revoked tickets in Barcelona would be upper-bounded by $512 \times 12 \times 110.529$ bytes ≈ 663 MB. Nevertheless, that is a rather loose upper bound and the actual storage would take much less space.

Chapter 6

Conclusions

In this thesis we have discussed privacy-preserving solutions for vehicular systems. More specifically, we have focused on pay-by-phone parking systems for public regulated areas and e-tickets that allow transfers between connecting lines. In all cases, we have analyzed the current proposals and we have designed protocols that improve the current state of the art.

Regarding parking systems, we have fulfilled two objectives. Our first objective was to design a system in which drivers perform a unique payment at the beginning of a parking transaction. As proposed in [BS19], the drivers pay for a set of tickets when they start the parking transaction. Tickets are paid with valued and no-valued e-coins to mask the expected duration of the parking transaction. These tickets cannot be linked among them. Also, the system server is unable to distinguish whether a given ticket has been paid with a valued or a no-valued e-coin. In case of inspection, the driver is required to provide the corresponding ticket which will be valid only if it was paid with a valued e-coin. Otherwise, the driver will be fined. The proposal guarantees the drivers' privacy. The second objective was to design a solution alternative to [BS19] providing a lower cost by avoiding the use of the cut-and-choose technique. This technique is computationally expensive because it requires the generation of a large amount of data which is to be discarded after the checking has been concluded. In [BS20b] we propose a novel system in which an e-coin contains a public key. The difference between a valued and a no-valued e-coin lies in whether or not the corresponding private key is known. When a ticket is generated, the system encrypts the ticket under the public key contained in the spent e-coin. If that e-coin is valued then the driver will be able to decrypt the ticket by using the private key.

Regarding e-ticket systems for transportation systems, we noticed a lack of proposals allowing e-ticket reusability. Hence, we aimed to design a construction allowing current e-ticket systems to be endowed with that property. So as to preserve users' privacy, the system had to be designed so that as a result of an inspection the system is only able to determine whether the inspected user is allowed to perform the current journey. In [BS20a] we propose a solution that can be coupled to any existing privacy-enabled ticketing system. After the purchase and validation of an e-ticket of the underlying system, our

proposal proceeds by generating a chain of *travel tokens*. In this way, if $\mathtt{T}$ journeys are allowed, a $\mathtt{T}$-element hash chain $\{chain_k\}_{1 \leq k \leq \mathtt{T}}$ is generated so that $chain_{k-1} = \mathcal{H}(chain_k)$, with $chain_{\mathtt{T}}$ being a random number. These *travel tokens* are released sequentially each time the user starts a new journey at the beginning of her trip or after performing a transfer. In addition, the *travel token* chain has an expiration time set during its activation. In order not to reveal the activation time, the protocol uses a zero-knowledge range proof of a committed value. The system is unable to link the *travel tokens* of a chain even after an inspection.

Future work

As future work, we plan to investigate the design of pay-by-phone parking systems in which the on-board device is able to provide by itself all the information required during a parking inspection. That is, the on-board device should be able to deliver the ticket to the officer when an inspection takes place.

We will also investigate other applications for e-coin systems providing valued and no-valued e-coins. Their features could probably lead to simple and efficient proposals for priced oblivious transfer protocols or secret sharing schemes.

Bibliography

[05] *Public Key Cryptography for the Financial Services Industry - the Elliptic Curve Digital Signature Algorithm (ECDSA): ANSI American National Standard for Financial Services, ANS X9.62-2005*. American national standard / ANSI. Accredited Standards Committee X9, Incorporated, 2005.

[19a] *Cambridge Dictionary*. 2019. URL: https://dictionary.cambridge.org/.

[19b] *First organized public transit system*. 2019. URL: https://www.wired.com/2008/03/march-18-1662-the-bus-starts-here-in-paris/.

[19c] *Tor Project*. 2019. URL: https://www.torproject.org.

[20a] *elparking*. 2020. URL: https://elparking.com.

[20b] *NIST SHA-3 Competition*. 2020. URL: https://csrc.nist.gov/projects/hash-functions/sha-3-project.

[20c] *Pango Mobile Parking*. 2020. URL: http://www.mypango.com.

[20d] *Parkmobile*. 2020. URL: http://www.parkmobile.com.

[20e] *ParkRight*. 2020. URL: https://www.westminster.gov.uk/parkright.

[20f] *PayByPhone*. 2020. URL: https://www.paybyphone.com.

[20g] *Paypal*. 2020. URL: https://www.paypal.com/.

[20h] *PayStay*. 2020. URL: https://paystay.com.au.

[20i] *RingGo*. 2020. URL: https://www.myringgo.co.uk.

[20j] *Telpark*. 2020. URL: https://www.telpark.com/.

[AAA17] Sattar J Aboud, Moustafa Al-Fayoumi and Mohammad Al-Fayoumi. 'Efficient e-Tickets fare scheme for time-typed services'. In: *Journal of Internet Banking and Commerce* 22.1 (2017).

[Ada+01] C. Adams, P. Cain, D. Pinkas and R. Zuccherato. *Internet X.509 Public Key Infrastructure Time-Stamp Protocol*. Tech. rep. IETF - Request For Comments RFC 3161, 2001.

[AF96] Masayuki Abe and Eiichiro Fujisaki. 'How to Date Blind Signatures'. In: *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology*. ASIACRYPT '96. Berlin, Heidelberg: Springer-Verlag, 1996, pp. 244–251. ISBN: 3540618724.

[AKJ10] A. S. Amoli, M. Kharrazi and R. Jalili. '2Ploc: Preserving Privacy in Location-based Services'. In: *2010 IEEE Second International Conference on Social Computing*. Aug. 2010, pp. 707–712.

[AO00] M. Abe and T. Okamoto. 'Provably Secure Partially Blind Signatures'. In: *Advances in Cryptology, EUROCRYPT 2000* (Aug. 2000), pp. 271–286.

[Aok+09] Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki and Lei Wang. 'Preimages for Step-Reduced SHA-2'. In: Dec. 2009, pp. 578–597. DOI: 10.1007/978-3-642-10366-7_34.

[Arf+14] Ghada Arfaoui, Guillaume Dabosville, Sébastien Gambs, Patrick Lacharme and Jean-François Lalande. 'A Privacy-Preserving NFC Mobile Pass for Transport Systems.' In: *ICST Trans. Mobile Communications Applications* 2.5 (2014), e4. DOI: 10.4108/mca.2.5.e4.

[Arf+15] Ghada Arfaoui, Jean-François Lalande, Jacques Traoré, Nicolas Desmoulins, Pascal Berthomé and Said Gharout. 'A Practical Set-Membership Proof for Privacy-Preserving NFC Mobile Ticketing'. In: *PoPETs* 2015 (2015), pp. 25–45.

[Asg11] Nabiha Asghar. *A survey on blind digital signatures*. Tech. rep. Department of Combinatorics & Optimization, University of Waterloo, ON, Canada, 2011.

[Bab34] Roger B. Babson. 'Automatic parking meter'. In: *US Patent 1973275A* (1934).

[BCK96a] M. Bellare, R. Canetti and H. Krawczyk. 'Keying hash functions for message authentication'. In: *Proc. of Crypto'96*. 1996, pp. 1–15.

[BCK96b] Mihir Bellare, Ran Canetti and Hugo Krawczyk. 'Pseudorandom functions revisited: The cascade construction and its concrete security'. In: *Proceedings of 37th Conference on Foundations of Computer Science*. IEEE. 1996, pp. 514–523.

[Bel+08] Bellare, Chanathip Namprempre, David Pointcheval and Semanko. 'The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme'. In: *Journal of Cryptology* 16 (Mar. 2008), pp. 185–215. DOI: 10.1007/s00145-002-0120-1.

[Ber+13] Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche. 'Keccak'. In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2013, pp. 313–314.

[BM15] A. W. Burange and H. D. Misalkar. 'Review of Internet of Things in development of smart cities with data management privacy'. In: *2015 International Conference on Advances in Computer Engineering and Applications*. 2015, pp. 189–195.

[BM16] R. Boyer and K. Mew. *Android Application Development Cookbook*. Packt Publishing, 2016.

[Bod+14] R. Bodenheimer, A. Brauer, D. Eckhoff and R. German. 'Enabling GLOSA for adaptive traffic lights'. In: *2014 IEEE Vehicular Networking Conference (VNC)*. 2014, pp. 167–174.

[Bou00] Fabrice Boudot. 'Efficient Proofs that a Committed Number Lies in an Interval'. In: *Advances in Cryptology, EUROCRYPT 2000* (Aug. 2000), pp. 431–444.

[BPV12] Olivier Blazy, David Pointcheval and Damien Vergnaud. 'Compact Round-Optimal Partially-Blind Signatures'. In: *Security and Cryptography for Networks*. Ed. by Ivan Visconti and Roberto De Prisco. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 95–112. ISBN: 978-3-642-32928-9.

[BR95] M. Bellare and P. Rogaway. 'Optimal asymmetric encryption'. In: *Advances in Cryptology — EUROCRYPT'94*. Ed. by Alfredo De Santis. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 92–111. ISBN: 978-3-540-44717-7.

[Bri+88] Ernest F. Brickell, David Chaum, Ivan B. Damgård and Jeroen van de Graaf. 'Gradual and Verifiable Release of a Secret (Extended Abstract)'. In: *Advances in Cryptology — CRYPTO '87*. Ed. by Carl Pomerance. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 156–166. ISBN: 978-3-540-48184-3.

[BS19] R. Borges and F. Sebé. 'Parking Tickets for Privacy-Preserving Pay-by-Phone Parking'. In: *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*. WPES'19. London, United Kingdom: ACM, 2019, pp. 130–134. ISBN: 978-1-4503-6830-8. DOI: 10.1145/3338498.3358638.

[BS20a] R. Borges and F. Sebé. 'A Construction for Providing Reusability to Mobile Phone-Based e-Tickets'. In: *IEEE Access* 8 (May 2020), pp. 101386–101397. DOI: 10.1109/ACCESS.2020.2998504.

[BS20b] R. Borges and F. Sebé. 'An efficient privacy-preserving pay-by-phone system for regulated parking areas'. In: *Int. J. Inf. Secur.* (Nov. 2020), pp. 1–13. DOI: 10.1007/s10207-020-00527-2.

[But14] Vitalik Buterin. 'Ethereum: A next-generation cryptocurrency and decentralized application platform'. In: *Bitcoin Magazine* 23 (2014).

[CDN09] Andrea Caragliu, Chiara Del Bo and Peter Nijkamp. 'Smart Cities in Europe'. In: *VU University Amsterdam, Faculty of Economics, Business Administration and Econometrics, Serie Research Memoranda* 18 (Jan. 2009). DOI: 10.1080/10630732.2011.601117.

[Cha83] David Chaum. 'Blind Signatures for Untraceable Payments'. In: *Advances in Cryptology*. Ed. by David Chaum, Ronald L. Rivest and Alan T. Sherman. Boston, MA: Springer US, 1983, pp. 199–203. ISBN: 978-1-4757-0602-4.

[Com18] European Commission. *EU General Data Protection Regulation*. 2018. URL: https://eugdpr.org/.

[CPS94] Jan Camenisch, Jean-Marc Piveteau and Markus Stadler. 'Blind Signatures Based on the Discrete Logarithm Problem'. In: *EUROCRYPT*. 1994.

[CYC04] Sherman Chow, Sm Yiu and K.P. Chow. 'Two Improved Partially Blind Signature Schemes from Bilinear Pairings'. In: 3574 (June 2004). DOI: 10.1007/b137750.

[CZL05] Xiaofeng Chen, Fangguo Zhang and Shengli Liu. 'ID-Based Restrictive Partially Blind Signatures'. In: vol. 3802. Dec. 2005, pp. 117–124. DOI: 10.1007/11596981_17.

[DDS12] Itai Dinur, Orr Dunkelman and Adi Shamir. 'New Attacks on Keccak-224 and Keccak-256'. In: *Fast Software Encryption*. Ed. by Anne Canteaut. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 442–461. ISBN: 978-3-642-34047-5.

[DDS14a] Itai Dinur, Orr Dunkelman and Adi Shamir. 'Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials'. In: July 2014, pp. 219–240. ISBN: 978-3-662-43932-6. DOI: 10.1007/978-3-662-43933-3_12.

[DDS14b] Itai Dinur, Orr Dunkelman and Adi Shamir. 'Improved Practical Attacks on Round-Reduced Keccak'. In: *Journal of Cryptology* 27 (Apr. 2014). DOI: 10.1007/s00145-012-9142-5.

[DEM15] Christoph Dobraunig, Maria Eichlseder and Florian Mendel. 'Analysis of SHA-512/224 and SHA-512/256'. In: Dec. 2015, pp. 612–630. DOI: 10.1007/978-3-662-48800-3_25.

[DH76] W. Diffie and M. Hellman. 'New directions in cryptography'. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.

[Dhu+15] D. Dhungana, G. Engelbrecht, J. X. Parreira, A. Schuster and D. Valerio. 'Aspern smart ICT: Data analytics and privacy challenges in a smart city'. In: *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. 2015, pp. 447–452.

[Din+15] Itai Dinur, Paweł Morawiecki, Josef Pieprzyk, Marian Srebrny and Michał Straus. 'Cube Attacks and Cube-Attack-Like Cryptanalysis on the Round-Reduced Keccak Sponge Function'. In: *Advances in Cryptology – EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 733–761. ISBN: 978-3-662-46800-5.

[DK00] William Daley and Raymond Kammer. 'Digital signature standard (DSS)'. In: (Jan. 2000), p. 77.

[ElG85] T. ElGamal. 'A public key cryptosystem and a signature scheme based on discrete logarithms'. In: *IEEE Transactions on Information Theory* 31.4 (1985), pp. 469–472.

[EW18] D. Eckhoff and I. Wagner. 'Privacy in the Smart City—Applications, Technologies, Challenges, and Solutions'. In: *IEEE Communications Surveys Tutorials* 20.1 (2018), pp. 489–516.

[FFS88] Uriel Feige, Amos Fiat and Adi Shamir. 'Zero-knowledge proofs of identity'. In: *Journal of Cryptology* 1 (June 1988), pp. 77–94. DOI: 10.1007/BF02351717.

[FL01] C. Fan and C. Lei. 'Cryptanalysis on improved user efficient blind signatures'. In: *Electronics Letters* 37.10 (2001), pp. 630–631.

[FL98] Chun-I Fan and Chin-Laung Lei. 'Low-Computation Partially Blind Signatures for Electronic Cash'. In: *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences* E81-A (Jan. 1998), pp. 818–824.

[FO98] Eiichiro Fujisaki and Tatsuaki Okamoto. 'A Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications'. In: *EUROCRYPT*. 1998.

[FPS19] Georg Fuchsbauer, Antoine Plouviez and Yannick Seurin. 'Blind Schnorr Signatures in the Algebraic Group Model'. In: *IACR Cryptology ePrint Archive* 2019 (2019), p. 877.

[Fuj+04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval and Jacques Stern. 'RSA-OAEP is secure under the RSA assumption'. In: *Journal of Cryptology* 17 (Mar. 2004), pp. 81–104. DOI: 10.1007/s00145-002-0204-y.

[Fuj+99] Ko Fujimura, Hiroshi Kuno, Masayuki Terada, Kazuo Matsuyama, Yasunao Mizuno and Jun Sekine. 'Digital-Ticket-Controlled Digital Ticket Circulation'. In: *Proceedings of the 8th USENIX Security Symposium* (1999), pp. 229–240.

[Gau07] Praveen Gauravaram. 'Cryptographic hash functions : cryptanalysis, design and applications'. In: 2007.

[GC88] D. Chaum G. Brassard and C. Crepeau. 'Minimum disclosure proofs of knowledge'. In: *Journal of Computer and System Sciences (37)* (1988), pp. 156–189.

[GHS10] Víctor Gayoso, Luis Hernandez and Carmen Sánchez. 'A Survey of the Elliptic Curve Integrated Encryption Scheme'. In: *Journal of Computer Science and Engineering* 2 (Jan. 2010), pp. 7–13.

[Gif+07] Rudolf Giffinger, Christian Fertner, Hans Kramar, Robert Kalasek, Nataša Milanović and Evert Meijers. *Smart cities - Ranking of European medium-sized cities*. Jan. 2007, pp. -.

[GLS16] Jian Guo, Meicheng Liu and Ling Song. 'Linear Structures: Applications to Cryptanalysis of Round-Reduced Keccak'. In: Dec. 2016, pp. 249–274. ISBN: 978-3-662-53886-9. DOI: 10.1007/978-3-662-53887-6_9.

[GMS17] R. Garra, S. Martínez and F. Sebé. 'A Privacy-Preserving Pay-by-Phone Parking System'. In: *IEEE Transactions on vehicular technology* 66.7 (2017), pp. 5697–5706.

[Gol01] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001.

[Guo+19] Jian Guo, Guohong Liao, Liu Guozhen, Meicheng Liu, Kexin Qiao and Ling Song. 'Practical Collision Attacks against Round-Reduced SHA-3'. In: *Journal of Cryptology* 33 (Feb. 2019). DOI: 10.1007/s00145-019-09313-3.

[Han+17] Jinguang Han, Liqun Chen, Steve Schneider, Helen Treharne and Steve Wesemeyer. 'Privacy-Preserving Electronic Ticket Scheme with Attribute-based Credentials'. In: *arXiv preprint arXiv:1706.03016* (2017).

[HC04] Hui-Feng Huang and Chin-Chen Chang. 'A new design of efficient partially blind signature scheme'. In: *Journal of Systems and Software* 73 (Nov. 2004), pp. 397–403. DOI: 10.1016/S0164-1212(03)00237-1.

[Hey+06] Thomas S. Heydt-Benjamin, Hee-Jin Chae, Benessa Defend and Kevin Fu. 'Privacy for Public Transportation'. In: *Privacy Enhancing Technologies*. Ed. by George Danezis and Philippe Golle. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–19. ISBN: 978-3-540-68793-1.

[HH08] Xiaoming Hu and Shangteng Huang. 'Analysis of ID-Based Restrictive Partially Blind Signatures and Applications'. In: *J. Syst. Softw.* 81.11 (Nov. 2008), pp. 1951–1954. ISSN: 0164-1212. DOI: 10.1016/j.jss.2008.01.013.

[HHR06] Iftach Haitner, Danny Harnik and Omer Reingold. 'Efficient Pseudorandom Generators from Exponentially Hard One-Way Functions'. In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone and Ingo Wegener. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 228–239. ISBN: 978-3-540-35908-1.

[HL01] Min-Shiang Hwang and Yuan-Liang Tang Yan-Chi Lai. 'Comment on "A Blind Signature Scheme Based On ElGamal Signature"'. In: *Technical Report CYUT-IM-TR2001-010, CYUT* (Aug. 2001).

[HLL02] Min-Shiang Hwang, CC Lee and YC Lai. 'Traceability on low-computation partially blind signatures for electronic cash'. In: *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences* E85A (May 2002), pp. 1181–1182.

[HM05] D. Hankerson and A. Menezes. 'Elliptic Curve Cryptography'. In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. van Tilborg. Boston, MA: Springer US, 2005, pp. 186–186. ISBN: 978-0-387-23483-0. DOI: 10.1007/0-387-23483-7_131.

[HS91] Stuart Haber and W. Scott Stornetta. 'How to time-stamp a digital document'. In: *Journal of Cryptology* 3 (1991), pp. 99–111.

[Hua+18] C. Huang, R. Lu, X. Lin and X. Shen. 'Secure automated valet parking: a privacy-preserving reservation scheme for autonomous vehicles'. In: *IEEE Transactions on Vehicular Technology* 67.11 (2018), pp. 11169–11180. DOI: 10.1109/TVT.2018.2870167.

[Inf20] Federal Office for Information Security. *Cryptographic Mechanisms: Recommendations and Key Lengths*. 2020. URL: https://www.bsi.bund.de.

[JLO97] Ari Juels, Michael Luby and Rafail Ostrovsky. 'Security of Blind Digital Signatures (Extended Abstract).' In: Jan. 1997, pp. 150–164.

[Kal92] Burton S. Kaliski. 'The MD2 Message-Digest Algorithm'. In: *RFC* 1319 (1992), pp. 1–17.

[KB00] Henryk Krawczyk and Mihir Bellare. 'HMAC: Keyed-Hashing for Message Authentication, RFC 2104'. In: 2000.

[KB15] T.E.W. Khattak and P. Buhler. *Big Data Fundamentals Concepts, Drivers and Techniques*. Prentice Hall, 2015.

[Kit13] R. Kitchin. 'The Real-Time City? Big Data and Smart Urbanism'. In: *Urban Research eJournal* (2013).

[KLG13] Florian Kerschbaum, Hoon Wei Lim and Ivan Gudymenko. 'Privacy-preserving billing for e-ticketing systems in public transportation'. In: *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM. 2013, pp. 143–154.

[Klí05] Vlastimil Klíma. 'Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications'. In: *IACR Cryptology ePrint Archive* 2005 (Jan. 2005), p. 102.

[KM16] Maninder Jeet Kaur and Piyush Maheshwari. 'Building smart cities applications using IoT and cloud-based architectures'. In: *2016 International Conference on Industrial Informatics and Computer Systems (CIICS)*. IEEE. 2016, pp. 1–5.

[Kob87] Neal Koblitz. 'Elliptic curve cryptosystems'. In: *Mathematics of computation* 48.177 (1987), pp. 203–209.

[LHH02] Cheng-Chi Lee, Li Hua and Min-Shiang Hwang. 'A remote user authentication scheme using hash functions'. In: *Operating Systems Review* 36 (Oct. 2002), pp. 23–29. DOI: 10.1145/583800.583803.

[Li+15] Yibin Li, Wenyun Dai, Zhong Ming and Meikang Qiu. 'Privacy Protection for Preventing Data Over-Collection in Smart City'. In: *IEEE Transactions on Computers* 65 (Jan. 2015), pp. 1–1. DOI: 10.1109/TC.2015.2470247.

[LL00] Chen-Ch Lin and Chi-Sung Laih. 'Cryptanalysis of Nyberg-Rueppel's message recovery scheme'. In: *Communications Letters, IEEE* 4 (Aug. 2000), pp. 231–232. DOI: 10.1109/4234.852925.

[LQP09] Yaohui Lei, Alejandro Quintero and Samuel Pierre. 'Mobile services access and payment through reusable tickets'. In: *Computer Communications* 32.4 (2009), pp. 602–610.

[LSK07] Jingwei Liu, Rong Sun and Weidong Kou. 'Fair E-payment protocol based on simple partially blind signature scheme'. In: *Wuhan University Journal of Natural Sciences* 12 (May 2007), pp. 181–184. DOI: 10.1007/s11859-006-0287-7.

[McK90] Kevin S. McKurley. 'The Discrete Logarithm Problem'. In: vol. 42. 1990.

[MEE00] Elsayed Mohammed, A.-E Emarah and K. El-Shennaway. 'A blind signature scheme based on ElGamal signature'. In: Feb. 2000, pp. C25/1–C25/6. ISBN: 977-5031-64-8. DOI: 10.1109/NRSC.2000.838954.

[Mil+12] M. Milutinovic, K. Decroix, V. Naessens and B. Decker. 'Privacy-Preserving Public Transport Ticketing System'. In: *Pierangela Samarati. 29th IFIP Annual Conference on Data and Applications Security and Privacy (DBSEC), Jul 2015, Fairfax, VA, United States. Springer International Publishing, Lecture Notes in Computer Science, LNCS-9149* D.95 (2012), pp. 135–150.

[Mil85] Victor S Miller. 'Use of elliptic curves in cryptography'. In: *Conference on the theory and application of cryptographic techniques*. Springer. 1985, pp. 417–426.

[Mon+13] C. Montes, D. Geldtmeijer, H. Rooden, J. Kohlmann, H. Slootweg, D. Van Leersum and M. Van Eekelen. 'A flexible, privacy enhanced and secured ICT architecture for a smart grid project with active consumers in the city of Zwolle — NL'. In: *22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013)*. 2013, pp. 1–4.

[MOV01] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001. URL: http://www.cacr.math.uwaterloo.ca/hac/.

[MSS98] Markus Michels, Markus Stadler and Hung -Min Sun. 'On the security of some variants of the RSA signature scheme'. In: *Computer Security — ESORICS 98*. Ed. by Jean-Jacques Quisquater, Yves Deswarte, Catherine Meadows and Dieter Gollmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 85–96. ISBN: 978-3-540-49784-4.

[Mut+12] Macià Mut-Puigserver, M. Magdalena Payeras-Capellà, Josep-Lluís Ferrer-Gomila, Arnau Vives-Guasch and Jordi Castellà-Roca. 'A survey of electronic ticketing applied to transport'. In: *Computers & Security* 31.8 (2012), pp. 925–939. DOI: 10.1016/j.cose.2012.07.004.

[Nak09] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2009. URL: http://www.bitcoin.org/bitcoin.pdf.

[Nat20a] United Nations. *The World's Cities in 2018*. 2020. URL: https://www.un.org/en/events/citiesday/assets/pdf/the_worlds_cities_in_2018_data_booklet.pdf.

[Nat20b] United Nations. *Universal Declaration of Human Rights*. 2020. URL: https://www.un.org/en/universal-declaration-human-rights/.

[NBS77] NBS. *FIPS 46. Data Encryption Standard. Federal Information Processing Standard*. U.S. Department of Commerce, Jan. 1977.

[NIS02] NIST. *Secure hash standard. Federal Information Processing Standard, FIPS-180-2*. U.S. Department of Commerce, Aug. 2002.

[NIS08] NIST. *NIST. Secure hash standard. Federal Information Processing Standard, FIPS-180-3*. U.S. Department of Commerce, Oct. 2008.

[NIS12] NIST. *NIST. Secure hash standard. Federal Information Processing Standard, FIPS-180-4*. U.S. Department of Commerce, Mar. 2012.

[NIS13] NIST. *Digital Signature Standard*. Tech. rep. National Institute of Standards and Technology, 2013.

[NIS15] NIST. *Secure hash standard. Federal Information Processing Standard, FIPS-202*. U.S. Department of Commerce, Aug. 2015.

[NIS93] NIST. *NIST. Secure hash standard. Federal Information Processing Standard, FIPS-180*. U.S. Department of Commerce, May 1993.

[NIS95] NIST. *Secure hash standard. Federal Information Processing Standard, FIPS-180-1*. U.S. Department of Commerce, Apr. 1995.

[NR93] Kaisa Nyberg and Rainer A. Rueppel. 'A New Signature Scheme Based on the DSA Giving Message Recovery'. In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. CCS '93. Fairfax, Virginia, USA: Association for Computing Machinery, 1993, pp. 58–61. ISBN: 0897916298. DOI: 10.1145/168588.168595. URL: https://doi.org/10.1145/168588.168595.

[Nua+15] Eiman Nuaimi, Hind Neyadi, Nader Mohamed and Jameela Al-Jaroodi. 'Applications of big data to smart cities'. In: *Journal of Internet Services and Applications* 6 (Aug. 2015). DOI: 10.1186/s13174-015-0041-5.

[NY89] Moni Naor and Moti Yung. 'Universal One-Way Hash Functions and their Cryptographic Applications'. In: *Proc. of the 21st STOC*. 1989, pp. 33–43.

[OS85] T. Okamoto and A. Shibaishi. 'A Fast Signature Scheme Based on Quadratic Inequalities'. In: *1985 IEEE Symposium on Security and Privacy*. 1985, pp. 123–123.

[PBS13] Pablo Pérez-Martínez, Antoni Ballesté and Agusti Solanas. 'Privacy in Smart Cities: A case study of smart public parking'. In: *PECCS 2013 - Proceedings of the 3rd International Conference on Pervasive Embedded Computing and Communication Systems* (Jan. 2013), pp. 55–59.

[PC97] Bhrat Patel and Jon Crowcroft. 'Ticket based service access for the mobile user'. In: *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking* (1997), pp. 223–233.

[Pér15] P.A. Pérez-Martínez. 'Contributions to Privacy Protection for Ubiquitous Computing'. Doctoral dissertation. Universitat Rovira i Virgili, 2015.

[Pet+15] J. Petit, F. Schaub, M. Feiri and F. Kargl. 'Pseudonym schemes in vehicular networks: a survey'. In: *IEEE Communication Surveys & Tutorials* 17.1 (2015), pp. 228–255. DOI: 10.1109/COMST.2014.2345420.

[Poi98] David Pointcheval. 'Strengthened security for blind signatures'. In: *Advances in Cryptology — EUROCRYPT'98*. Ed. by Kaisa Nyberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 391–405. ISBN: 978-3-540-69795-4.

[PR20] Pascual Berrone Pascual and Joan Enric Ricart. *IESE Cities in Motion Index 2020*. 2020. URL: https://media.iese.edu/research/pdfs/ST-0542-E.pdf.

[Pro20] Litecoin Project. *Litecoin*. 2020. URL: https://litecoin.org/.

[PS96] David Pointcheval and Jacques Stern. 'Provably secure blind signature schemes'. In: *Advances in Cryptology — ASIACRYPT '96*. Ed. by Kwangjo Kim and Tsutomu Matsumoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 252–265. ISBN: 978-3-540-70707-3.

[PSW12] Thomas Peyrin, Yu Sasaki and Lei Wang. 'Generic Related-Key Attacks for HMAC'. In: *Advances in Cryptology – ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 580–597. ISBN: 978-3-642-34961-4.

[QH05] D. Quercia and S. Hailes. 'MOTET: Mobile Transactions using Electronic Tickets'. In: *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*. Sept. 2005, pp. 374–383.

[Rab78] Michael O Rabin. 'Digitalized signatures'. In: *Foundations of secure computation* (1978), pp. 155–168.

[RD92] Ronald Rivest and S Dusse. *The MD5 message-digest algorithm*. 1992.

[Res18] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Aug. 2018. DOI: 10.17487/RFC8446. URL: https://rfc-editor.org/rfc/rfc8446.txt.

[Riv91] Ronald L. Rivest. 'The MD4 Message Digest Algorithm'. In: *Advances in Cryptology-CRYPTO' 90*. Ed. by Alfred J. Menezes and Scott A. Vanstone. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 303–311. ISBN: 978-3-540-38424-3.

[RK13] K. Richter and J. Keeley. *iOS Components and Frameworks: Understanding the Advanced Features of the iOS SDK*. Addison Wesley, 2013.

[RSA78] R. Rivest, A. Shamir and L. Adleman. 'A method for obtaining digital signatures and public-key cryptosystems'. In: *Communications of the ACM* 21 (1978), pp. 120–126.

[Rup+13] Andy Rupp, Gesine Hinterwälder, Foteini Baldimtsi and Christof Paar. 'P4R: Privacy-Preserving Pre-Payments with Refunds for Transportation Systems'. In: *Financial Cryptography and Data Security*. Ed. by Ahmad-Reza Sadeghi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 205–212.

[Sch+08] Elmar Schoch, Frank Kargl, Michael Weber and Tim Leinmüller. 'Communication patterns in VANETs'. In: *Communications Magazine, IEEE* 46 (Dec. 2008), pp. 119–125. DOI: 10.1109/MCOM.2008.4689254.

[Sch05] Paul Schwartz. 'Property, Privacy, and Personal Data'. In: *Harvard Law Review* 117 (May 2005). DOI: 10.2307/4093335.

[Sch20] Berry Schoenmakers. *Lecture Notes Cryptographic Protocols*. Technical University of Eindhoven, Feb. 2020.

[Sch82] Claus-Peter Schnorr. 'Refined analysis and improvements on some factoring algorithms'. In: *Journal of Algorithms* 3.2 (1982), pp. 101–127.

[Sch90] C. P. Schnorr. 'Efficient Identification and Signatures for Smart Cards'. In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Ed. by Gilles Brassard. New York, NY: Springer New York, 1990, pp. 239–252. ISBN: 978-0-387-34805-6.

[Sch96] Bruce Schneier. *Applied Cryptography, Second Edition: Protocols, Algorithms and Source Code in C*. John Wiley & Sons, 1996.

[SG12] Rajeev Sobti and Geetha Ganesan. 'Cryptographic Hash Functions: A Review'. In: *International Journal of Computer Science Issues, ISSN (Online): 1694-0814* Vol 9 (Mar. 2012), pp. 461–479.

[Sha00] Zuhua Shao. 'Improved user efficient blind signatures'. In: *Electronics Letters* 36.16 (2000), pp. 1372–1374.

[SS08] Somitra Kumar Sanadhya and Palash Sarkar. 'New Collision Attacks against Up to 24-Step SHA-2'. In: *Progress in Cryptology - INDOCRYPT 2008*. Ed. by Dipanwita Roy Chowdhury, Vincent Rijmen and Abhijit Das. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 91–103. ISBN: 978-3-540-89754-5.

[SSA17] Ahmad Steef, M. N. Shamma and A. Alkhatib. 'A secure approach for embedding message text on an elliptic curve defined over prime fields, and building 'EC-RSA-ELGamal' Cryptographic System'. In: *ArXiv* abs/1707.04892 (2017).

[Ste06] Marc Stevens. 'Fast Collision Attack on MD5'. In: *IACR Cryptology ePrint Archive* 2006 (Jan. 2006), p. 104.

[SVW08] Ahmad-Reza Sadeghi, Ivan Visconti and Christian Wachsmann. 'User privacy in transport systems based on RFID e-tickets'. In: *PiLBA'08 Privacy in Location-Based Applications* (2008), pp. 102–121.

[TIA18] Nedal Tahat, Eddie Shahril Ismail and A.K. Alomari. 'Partially blind signature scheme based on chaotic maps and factoring problems'. In: *Italian Journal of Pure and Applied Mathematics* (Feb. 2018), pp. 165–177.

[Tra06] Wade Trappe. *Introduction to cryptography with coding theory*. Pearson Education India, 2006.

[TSI08] Nedal Tahat, Shatnawi S.M.A and Eddie Shahril Ismail. 'A New Partially Blind Signature Based on Factoring and Discrete Logarithms'. In: *Journal of Mathematics and Statistics* 4 (Feb. 2008). DOI: 10.3844/jmssp.2008.124.129.

[Ver+08] Kristof Verslype, Bart De Decker, Vincent Naessens, Girma Nigusse, Jorn Lapon and Pieter Verhaeghe. 'A Privacy-Preserving Ticketing System'. In: *Data and Applications Security XXII*. Ed. by Vijay Atluri. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 97–112. ISBN: 978-3-540-70567-3.

[Viv+11] 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 and Autonomous Spontaneous Security*. Ed. by Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, Ana Cavalli and Jean Leneutre. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 79–92. ISBN: 978-3-642-19348-4.

[Viv+12] A. Vives, M. Payeras, M. Mut, J. Castella and J. Ferrer. 'A Secure E-Ticketing Scheme for Mobile Devices with Near Field Communication (NFC) That Includes Exculpability and Reusability'. In: *IEICE TRANSACTIONS on Information and Systems* D.95 (2012), pp. 78–93.

[Wag02] David Wagner. 'A Generalized Birthday Problem'. In: *Advances in Cryptology — CRYPTO 2002*. Ed. by Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 288–304. ISBN: 978-3-540-45708-4.

[Wan+04a] Hua Wang, Yanchun Zhang, Jinli Cao and Yahiko Kambayahsi. 'A Global Ticket-Based Access Scheme for Mobile Users'. In: *Information Systems Frontiers* 6.1 (2004), pp. 35–46. ISSN: 1572-9419.

[Wan+04b] Xiaoyun Wang, Dengguo Feng, Xuejia Lai and Hongbo Yu. 'Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.' In: *IACR Cryptology ePrint Archive* 2004 (Jan. 2004), p. 199.

[WYY05a] Xiaoyun Wang, Yiqun Lisa Yin and Hongbo Yu. 'Finding Collisions in the Full SHA-1'. In: *Advances in Cryptology – CRYPTO 2005*. Ed. by Victor Shoup. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 17–36. ISBN: 978-3-540-31870-5.

[WYY05b] Xiaoyun Wang, Hongbo Yu and Yiqun Yin. 'Efficient Collision Search Attacks on SHA-0'. In: vol. 3621. Aug. 2005, pp. 1–16. ISBN: 978-3-540-28114-6. DOI: 10.1007/11535218_1.

[Yan+11] G. Yan, W. Yang, D.B. Rawat and S. Olariu. 'SmartParking: a secure and intelligent parking system'. In: *IEEE Intelligent Transportation Systems Magazine* 3.1 (2011), pp. 18–30. DOI: 10.1109/MITS.2011.940473.

[Yan+13] York Yannikos, Jonathan Schlüßler, Martin Steinebach, Christian Winter and Kalman Graffi. 'Hash-Based File Content Identification Using Distributed Systems'. In: vol. 410. Jan. 2013. DOI: 10.1007/978-3-642-41148-9_8.

[ZC05] Fangguo Zhang and Xiaofeng Chen. 'Cryptanalysis of Huang–Chang partially blind signature scheme'. In: *Journal of Systems and Software* 76 (June 2005), pp. 323–325. DOI: 10.1016/j.jss.2004.07.249.

[Zha+15] Y. Zhang, C.C. Tan, F. Xu, H. Han and Q. Li. 'VProof: lightweight privacy-preserving vehicle location proofs'. In: *IEEE Transactions on Vehicular Technology* 64.1 (2015), pp. 378–385. DOI: 10.1109/TVT.2014.2321666.

[ZPS93] Yuliang Zheng, Josef Pieprzyk and Jennifer Seberry. 'HAVAL — A one-way hashing algorithm with variable length of output (extended abstract)'. In: *Advances in Cryptology — AUSCRYPT '92*. Ed. by Jennifer Seberry and Yuliang Zheng. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 81–104. ISBN: 978-3-540-47976-5.