CRYPTOGRAPHIC PROTOCOLS FOR PRIVACY-AWARE AND SECURE E-COMMERCE

Alberto Blanco Justicia

UNIVERSITAT ROVIRA i VIRGILI

Cryptographic Protocols for Privacy-Aware and Secure e-Commerce

Alberto Blanco-Justicia

DOCTORAL THESIS

2017

Alberto Blanco-Justicia

Cryptographic Protocols for Privacy-Aware and Secure e-Commerce

DOCTORAL THESIS

Supervised by Dr. Josep Domingo-Ferrer

Department of Computer Engineering and Mathematics

UNIVERSITAT ROVIRA i VIRGILI

January, 2017

I STATE that the present study, entitled "Cryptographic Protocols for Privacy-Aware and Secure e-Commerce", presented by Alberto Blanco Justicia for the award of the degree of Doctor, has been carried out under my supervision at the Department of Computer Engineering and Mathematics of this university, and that it fulfills all the requirements to be eligible for the International Doctorate Award.

Tarragona, February 8, 2017

Doctoral Thesis Supervisor

Prof. Dr. Josep Domingo-Ferrer

Acknowledgments

I would like to thank my advisor Prof. Josep Domingo-Ferrer for his guidance during the development of this thesis. The guidance of Dr. Carla Ràfols, Dr. Oriol Farràs and Prof. Qianhong Wu is also gratefully acknowledged, regarding respectively the IBDT signature scheme, private set intersection, and implicit authentication using Paillier. I am also indebted to all CRISES group members, especially to Jesús Manjón. Thanks to my parents, sister and niece for their unconditional support and love. Also to all my friends for always being there. Finally, I am grateful to Sven Rosinger, Stefan Janacek and the rest of the OFFIS team, for their hospitality during my stay in Oldenburg (Germany) in spring 2016.

This work was partly funded by Google through a Google Faculty Research Award to Prof. Josep Domingo-Ferrer; by the Government of Catalonia under grants 2009 SGR 1135, 2014 SGR 537 and FI-DGR 00207; by the Spanish Government through projects TIN2011-27076-C03-01 "CO-PRIVACY", CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES", and TIN2014-57364-C2-R "SmartGlacis"; and by the European Commission under FP7 projects "DwB" and "Inter-Trust" and H2020 projects "CLARUS" and "CANVAS".

Contents

List of Figures

List of Tables

Abstract

The latest Eurobarometer published in December 2016, reflecting the perceptions of the European citizens on privacy and security in telecommunications, shows that, although people are not always informed on the privacy regulations or the implications of privacy breaches, they demand specific privacy protection. In particular, citizens want their data, their communications and the data that they give or outsource to online services to be well protected and not shared with unwanted parties. The demands of the public can be partly covered by the application of privacy-by-design principle and the use of Privacy Enhancing Techniques (PETs) in commercial applications. The use of PETs by major service providers is not widespread, as they are typically seen as negatively impacting on the performance of applications and the functionalities they can offer. Moreover, major providers use the data they obtain from their customers (often more data than strictly needed to run the service being offered) to sell targeted advertising services to third parties (usually smaller service providers or application developers that do not have enough data of their own). In this work we combine known and novel privacy-enhancing techniques to design new commercial applications and services that respect and protect the identity of the customers and/or their sensitive data. The applications we present are group discounts, loyalty programs and implicit authentication. For group discounts, we propose a novel privacy-preserving group size accreditation method, whereby a service provider can cryptographically verify the number of members of a group accessing a service, while the group members remain anonymous. Regarding loyalty programs, we propose a novel privacy-preserving loyalty program construction whereby a service provider can issue and verify loyalty points, as well as profile users to the level chosen by the users themselves: the user is rewarded with more loyalty points as she reveals more details on her purchase history. In what regards implicit authentication, we propose two new mechanisms based on secure multiparty computation that make it possible to conduct

authentication based on the user's profile while protecting the privacy of the profile. The privacy-preserving features of the mechanisms proposed in this thesis have a quantifiable and limited impact on the security and performance of applications.

Resum

El darrer Eurobaròmetre, publicat al desembre de 2016, referent a les percepcions dels ciutadans europeus pel que fa a la privadesa i a la seguretat en telecomunicacions, mostra que, encara que els ciutadans no sempre estiguin al corrent de les normes sobre privadesa ni de la rellevància de les violacions de la privadesa, demanen protecció. Concretament, els ciutadans volen que llurs dades, llurs comunicacions i les dades que forneixen als serveis en línia estiguin ben protegides i no es comparteixin amb tercers. Aquestes demandes de la ciutadania es poden satisfer en part aplicant el principi de privadesa per disseny i fent servir tècniques de millora de la privadesa (PETs, de l'anglès *Privacy Enhancing Techniques*) en aplicacions comercials. L'ús de PETs per part dels grans proveïdors de serveis no és gaire comú, car se sol pensar que aquest tipus de tècniques tenen un impacte negatiu en el rendiment i en les funcionalitats de les aplicacions. A banda, els grans proveïdors aprofiten les dades que recullen de llurs usuaris (sovint més dades de les estrictament necessàries per al servei que ofereixen) per vendre serveis de màrqueting a tercers (normalment petits proveïdors o desenvolupadors independents d'aplicacions que no tenen prou dades ells mateixos). En aquest treball, apliquem tècniques conegudes i també tècniques noves de millora de privadesa al disseny de noves aplicacions i serveis comercials que respecten i protegeixen la identitat dels usuaris i/o llurs dades sensibles. Les aplicacions que considerem són descomptes de grup, programes de fidelització, i autenticació implícita. Pel que fa als descomptes de grup, proposem un mecanisme d'acreditació de la mida d'un grup amb preservació de privadesa, mitjançant el qual el proveïdor d'un servei pot comprovar criptogràficament el nombre de membres d'un grup que vol accedir al servei, alhora que els membres preserven llur anonimat. Pel que fa als programes de fidelització, en proposem una construcció respectuosa amb la privadesa, amb la qual el proveïdor pot emetre i verificar punts de fidelització, així com perfilar els usuaris en la mesura que aquests ho permetin: l'usuari pot triar el nivell

de perfilatge i és recompensat amb més punts de fidelització com més detalls revela d'allò que ha comprat. Pel que fa a l'autenticació implícita, proposem dos nous mecanismes basats en protocols de computació multipart, que permeten d'autenticar l'usuari a partir del seu perfil tot i protegint la privadesa del perfil. Les propietats de preservació de la privadesa dels mecanismes proposats en aquesta tesi tenen un impacte quantificable i limitat sobre la seguretat i el rendiment de les aplicacions.

Resumen

El último Eurobarómetro, publicado en diciembre de 2016, referido a la percepción de los ciudadanos europeos respecto a la privacidad y seguridad de las telecomunicaciones, muestra que, aunque los ciudadanos no siempre estén al corriente de la normativa sobre privacidad ni de la relevancia de las violaciones de la privacidad, piden protección. En concreto, los ciudadanos quieren que sus datos, sus comunicaciones y los datos que suministran a los servicios en línea estén bien protegidos y no se compartan con terceros. Estas demandas de la ciudadanía podrían resolverse parcialmente aplicando el principio de privacidad por diseño y usando técnicas de mejora de la privacidad (PETs, del inglés *Privacy Enhancing Techniques*) en aplicaciones comerciales. El uso de PETs por parte de los grandes proveedores de servicios no es muy común, ya que se considera que este tipo de técnicas impactan negativamente en el rendimiento de las aplicaciones y limitan sus funcionalidades. Además, los grandes proveedores utilizan habitualmente los datos obtenidos de los usuarios, tanto aquellos imprescindibles para el funcionamiento de la aplicación como aquellos que no lo son, para ofrecer servicios de márqueting a terceros (con frecuencia, pequeños proveedores de servicios o desarrolladores independientes de aplicaciones que no suelen tener bastantes datos ellos mismos). En este trabajo, aplicamos técnicas conocidas y también técnicas nuevas de mejora de privacidad al diseño de nuevas aplicaciones y servicios comerciales que respetan y protegen la identidad de los usuarios y/o sus datos sensibles. Las aplicaciones consideradas son descuentos de grupo, programas de fidelización, y autenticación implícita. En cuanto a descuentos de grupo, proponemos un mecanismo de acreditación del tamaño de un grupo con preservación de privacidad, mediante el cual el proveedor de un servicio puede comprobar criptográficamente el número de miembros de un grupo que quiere acceder al servicio, a la vez que los miembros preservan su anonimato. En cuanto a los programas de fidelización, proponemos una nueva construcción respetuosa con la privacidad, con la cual el proveedor puede emitir

y verificar puntos de fidelización, así como perfilar a los usuarios en la medida que estos lo permitan: el usuario puede escoger el nivel de perfilado y es recompensado con más puntos de fidelización si revela más detalles sobre lo que ha comprado. En cuanto a autenticación implícita, proponemos dos mecanismos basados en protocolos de computación multiparte, que permiten autenticar al usuario a partir de su perfil, a la vez que se protege la privacidad del perfil. Las propiedades de privacidad de los mecanismos propuestos en esta tesis tienen un impacto cuantificable y limitado sobre la seguridad y el rendimiento de las aplicaciones.

Chapter 1

Introduction

1.1 Motivation

The latest Eurobarometer [39] reflecting the views of European citizens on security, privacy and surveillance of online communications shows that a majority of interviewed citizens are concerned about the privacy of their communications. Although most people incorrectly believe that the current laws completely prevent access to their private communications by companies and public organisms, they agree that there should be specific tools and measures to protect their privacy. This concern about privacy has increased in the few years since Snowden's revelations, that confirmed that the US government (and other governments as disclosed afterwards), conducted mass surveillance operations on their citizens on account of national security. Most developers of instant messaging applications, such as WhatsApp, have since deployed end-to-end encryption to meet the users' demands, but contextual data (a.k.a. metadata) are still being recorded by most online services and applications.

The privacy-by-design principle requires application designers to gather only the personal data that are essential to the correct operation of their applications. That is, applications following this principle should only ask the users to input those personal data that the specific application explicitly needs. Most applications in the market (and especially smartphone applications) clearly disregard this principle —a look at the permissions they request is conclusive—: they collect contextual information too, even if it is not needed. The privacy policies of major service providers explain that all data may be used for commercial

purposes. Although major service providers collect more data than strictly necessary, the privacy-by-design principle demands that users be empowered with the decision about when to grant access to their data, when to modify them and when to delete them. While this is more or less being taken into account by service providers, sometimes the procedures to modify or delete personal data are not transparent enough or too cumbersome.

Privacy Enhancing Techniques (PET) are cryptographic and non-cryptographic tools that, when used appropriately, minimize the amount of personal data being handled by applications, and therefore help developers to more easily comply with regulations on personal data processing. Therefore, research on privacy enhancing techniques and on the practical deployment of the privacy-by-design principle is backed by the demands of the general population.

In this work we aim at demonstrating that, if appropriate techniques are used, privacy does not necessarily work against security and/or utility. We focus on three specific application cases described below:

Group size accreditation mechanism. Group discounts are offered by vendors and public authorities to encourage a more sustainable (or profitable) way to access their services or use public resources. An example of this are high-occupancy vehicle (HOV) tolls in highways, which offer discounts for vehicles carrying more than a given number of passengers (2 or more, 3 or more, etc). There are several ways to ascertain the number of members of a group: employees at access points that count them, cameras that take photos and analyze them in toll booths, or registration procedures that require the names of all members of groups, among others. We argue that automated mechanisms, such as cameras and registration procedures, take more information from the participants that is actually needed (thus violating the privacy-by-design principle), and that the only really necessary information is the size of the groups.

Loyalty programs. Loyalty programs are marketing efforts implemented by vendors, especially retailers, that are aimed at establishing a lasting relationship with consumers. In a loyalty program, the vendor pursues two main goals: i) to encourage the consumer to make more purchases in the future (returning customer); ii) to allow the vendor to profile the consumer in view of conducting market research and segmentation (profiled customer). In order to lure consumers into a loyalty program, the vendor offers them rewards, typically loyalty points that consumers can later exchange for discounts, gifts or other benefits offered by the vendor. Normally, enrollment to loyalty programs involves some kind of registration

procedure, in which customers fill out a form with their personal information and are granted a loyalty card, be it a physical card (magnetic stripe or smartcard) or a smartphone application. Although loyalty programs have become widespread, they are experiencing a loss of active participants and they have been criticized by business experts and consumer associations. Criticism is mainly due to privacy issues, because it is not always clear whether the benefits offered by vendors in their loyalty programs are worth the loss of consumer privacy caused by profiling [78, 91, 1, 45].

Implicit authentication. Implicit authentication refers to a software system authenticating individuals based on the way they interact with their device, i.e. their behavior. In this context, the user's behavior can be determined by collecting a variety of features, such as keystroke patterns, browser history and configuration, IP addresses, location, visible antennas, etc. Implicit authentication can be viewed as a complement of the usual explicit authentication based on identifiers and credentials.

Note that a common trait in these three application cases is that users need to prove something about themselves or their context without revealing more than what is strictly necessary. We believe these cases can be used as an example for other applications in which the goal is similar.

1.2 Contributions

Our contributions are protocols to solve the privacy, security and functionality conflict in the above three applications. Specifically:

1. In the group size accreditation case (motivated by applications involving group discounts), we contribute a mechanism, based on an identity-based dynamic-threshold (IBDT) signature scheme and a novel key management scheme, that allows participants to prove the number of members in a group without revealing the identities of group members. The IBDT signature scheme is a novel advanced signature scheme that combines the properties of identity-based signatures and threshold signatures. In identity-based signatures, the public keys are arbitrary strings; we leverage this property to introduce a novel parameterized key management scheme, in which the functionality of the scheme (the size of the groups that can be certified) and the level of anonymity are defined by two configurable parameters. In threshold signature schemes, a key dealer distributes key

shares that, when combined in a specific way and in a specific quantity, produce signing keys that are valid for the system. The system is evaluated analytically and experimentally, namely in a high-occupancy vehicle application.

2. In the loyalty programs case, we contribute a mechanism, based on partially blind signatures and zero-knowledge proofs, that allows implementing loyalty programs, including loyalty points and customer profiling, and also allows customers to control the amount of sensitive information (from their purchase histories) they reveal. To that end, we entrust the management of loyalty points and purchase receipts to the customers themselves, who can generalize the purchase receipts before returning them to the vendor for more loyalty points (or not return them, at the cost of receiving less loyalty points). To ensure that loyalty points and purchase receipts are not altered by malicious participants, these are treated as electronic cash, but without a centralized authority. To prevent the vendors from linking loyalty points and purchase histories to specific customers, we introduce (untransferable) anonymous tokens with controlled linkability (by leveraging partially blind signatures and zero-knowledge proofs) and a generalization scheme for purchase receipts. The security of the scheme is formally evaluated and experimental results are provided. We finish this contribution by providing a feasibility study in physical stores.

3. In the implicit authentication case, we propose two schemes following the same principle of authenticating users based on their behavior without revealing the user profiles to the service provider. Both schemes are predicated on computing the distances between private feature sets. We show that these distances can be obtained from the size of the intersection of the feature sets, for sets of categorical independent values, categorical correlated values and numerical independent values. The computation of the size of the intersection of sets is carried out using secure multiparty computation (MPC) protocols. The first of our proposed implicit authentication schemes uses an MPC based on the homomorphic properties of the Paillier cryptosystem, and provides a robust solution for authentication. The second one is based on the intersection of Bloom filters. While the second proposal does not ensure as high a level of security as the first one (namely it does not provide semantic security), its lower computational complexity and the compact form of the protected user profiles make it suitable for everyday use (always as a second-factor authentication mechanism). We

provide formal security analyses and experimental evaluation of the two proposed implicit authentication schemes.

The first and second contributions of this thesis (privacy-preserving group size accreditation and privacy-preserving loyalty programs) received partial support from a Google Faculty Award granted to Prof. Josep Domingo-Ferrer. The third contribution (privacy-preserving implicit authentication) is an offshoot of the research started in the EU FP7 project "InterTrust".

Chapter 2

Background

In this Chapter we provide the background on cryptographic and non-cryptographic technologies that will be used as building blocks for our privacy-preserving constructions. The Chapter is organized as follows. Section 2.1 provides an overview of bilinear pairings and introduces some hardness assumptions that will be used in our group accreditation and loyalty programs constructions. Sections 2.2 and 2.3 introduce, respectively, threshold and identity-based signatures. These kinds of signatures are the main building blocks of our novel IBDT signature scheme (described in Chapter 3) on which our group accreditation protocol is based. Section 2.4 introduces the concept of partially blind signatures, a generalization of blind signatures which eliminates the need for cut-and-choose protocols and that is the main building block of our loyalty programs construction (described in Chapter 4). Section 2.5 provides an overview of zero-knowledge proofs, which we explicitly use in an extension to the loyalty programs construction to provide untransferability of loyalty points and purchase receipts. Non-interactive zero-knowledge proofs are also part of the IBDT signature scheme. Section 2.6 recalls secure multiparty computation protocols to compute set intersections. The size of the intersection of sets is the basis of our two proposed implicit authentication schemes. Sections 2.7 and 2.8 describe the Paillier cryptosystem and Bloom filters, which we use in our implicit authentication schemes to instantiate two concrete multiparty computation protocols to compute the size of the intersection of sets. Finally, Sections 2.9 and 2.10 briefly describe anonymous payment mechanisms and short-range communication technologies. These two are used when implementing the group accreditation protocol and the loyalty programs scheme in specific use cases.

2.1 Bilinear pairings

Given cyclic multiplicative groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ of prime order p, with generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, a bilinear map is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

- Bilinearity: For all $x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$, $a, b \in \mathbb{Z}_p$, $e(x^a, y^b) = e(x, y)^{ab}$.
- Non-degeneracy: The value $e(g_1, g_2)$ generates $\mathbb{G}_T$, that is $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$.
- Efficient computability: The function e is efficiently computable.

Bilinear pairings are classified in three general types according to [50]:

Type I: $\mathbb{G}_1 = \mathbb{G}_2$. These pairings are typically called symmetric pairings.

Type II: $\mathbb{G}_1 \neq \mathbb{G}_2$, and there exists an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ that is efficiently computable, but ψ^{-1} is not.

Type III: $\mathbb{G}_1 \neq \mathbb{G}_2$, and there is no efficiently computable ψ.

Types II and III are called asymmetric pairings. We will use Type III pairings throughout this work, as they have been found to be the most efficient pairings at equivalent security levels. We use multiplicative notation for all groups $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$.

2.1.1 Hardness assumptions

Definition 1. *Given a cyclic group $\mathbb{G}$ of order p and a generator g of $\mathbb{G}$, the* Computational Diffie-Hellman (CDH) *problem is defined as follows: given a tuple (g, g^α, g^γ), compute $g^{\alpha\gamma}$.*

Definition 2. *Given a cyclic group $\mathbb{G}$ of order p and a generator g of $\mathbb{G}$, the* Decisional Diffie-Hellman (DDH) *problem is defined as follows: given a tuple $(g, g^\alpha, g^\gamma, g^\delta)$, decide whether $\alpha\gamma = \delta$.*

Definition 3. *Given a cyclic group $\mathbb{G}$ of order p and a generator g of $\mathbb{G}$, the* chosen-target Computational Diffie-Hellman (chosen-target CDH) *problem [19] is defined as follows: given set $Z = \{z_1, \ldots, z_n\} \in \mathbb{G}^n$ a random public key $y = g^x$ and access to a helper oracle $(\cdot)^x$, output a set $V = \{v_1, \ldots, v_l\} \in \mathbb{G}^l$, such that for all $i \in \{1, \ldots, l\}$ there exists a $z_j \in Z$ with $z_j = v_i^x$, with less than l queries to the helper oracle. This problem is equivalent to the CDH problem if no access to the helper oracle is given.*

Definition 4. *In an asymmetric bilinear pairing* $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ *of prime order* p*, the* asymmetric n-Diffie-Hellman Exponent *(asymmetric* n*-DHE) problem [22] is defined as follows: given a tuple*

$$(g_1, g_1^{\gamma}, g_1^{\gamma^2}, \ldots, g_1^{\gamma^n}, g_1^{\gamma^{n+2}}, \ldots, g_1^{\gamma^{2n}}, g_2, g_2^{\gamma}, g_2^{\gamma^2}, \ldots, g_2^{\gamma^n})$$

where $\gamma \leftarrow \mathbb{Z}_p$*,* $g_1 \leftarrow \mathbb{G}_1$*,* $g_2 \leftarrow \mathbb{G}_2$*, compute* $g_1^{\gamma^{n+1}}$*.*

2.2 Threshold signature schemes

Threshold signature schemes are commonly based on (t, n)-threshold secret sharing schemes, such as the ones introduced in [12] and [86], and they require a minimum number t of participants to produce a valid signature.

A threshold signature scheme is defined as follows:

Definition 5. *(Threshold signature scheme) A threshold signature scheme consists of tree participants: trusted entity, user and verifier. There are five algorithms: Setup, KeyGen, Sign, Combine, and Verify.*

- *The Setup algorithm is a PPT algorithm that takes as input a security parameter* λ *and parameters* n *and* t*, and outputs the system parameters* pms*, a master secret key* msk *and a master public key* mpk*.*
- *The KeyGen algorithm is a PPT algorithm that takes as inputs the system public parameters* pms *and the master secret key* msk*, and outputs a secret key share* sk_i*.*
- *The Sign algorithm is a PPT algorithm that, given the system public parameters* pms*, a secret key* sk_i *and a message* m*, outputs a partial signature* σ_i*.*
- *The Combine algorithm is a PPT algorithm that, given a message* m *and a vector of partial signatures* $\{\sigma_i\}_{i=0}^{t}$ *on the message* m *outputs a signature* σ*.*
- *The Verify algorithm is a polynomial-time algorithm that takes as inputs the public parameters* pms*, a master public key* mpk*, a message* m *and a signature* σ*, and outputs* 1 *if* σ *is a valid signature on the message* m*;* 0 *otherwise.* σ *is a valid signature if at least* t *out of* n *partial signatures have been used to compute it.*

Our group accreditation mechanism makes use of a signature scheme that has a dynamic threshold. Dynamic threshold signature schemes differ from the previous ones in that the threshold t is not fixed during the set-up phase, but is declared at the time of computing the signature.

2.3 Identity-based signature schemes

Identity-based signature schemes, theorized by Shamir in [87] and with the first concrete protocol, based on the Weil pairing, developed by Boneh *et al.* in [23], allow public keys pk_{id} to be arbitrary strings of some length, which are called identities. These strings are associated with a user U and reflect some aspect of his identity, *e.g.* his email address. The corresponding secret key sk_{id} is then computed by a trusted entity, the certification authority (CA), taking as input the user's identity and, possibly, some secret information held only by the CA, and is sent to the user U through some secure channel. Identity-based public key signature schemes offer a great flexibility in key generation and management, which is a feature we require in our group size accreditation method.

An identity-based signature scheme is defined as follows:

Definition 6 (Identity-based signature scheme)**.** *An identity-based signature scheme consists of tree participants: trusted entity, user and verifier. There are four algorithms: Setup, Extract, Sign, and Verify.*

- *The Setup algorithm is a PPT algorithm that takes as input a security parameter λ and outputs the system parameters* pms *and a master secret key* msk*.*
- *The Extract algorithm is a PPT algorithm that takes as inputs the system public parameters* pms*, the master secret key* msk *and an identity id, and outputs a key-pair (pk_{id}, sk_{id}).*
- *The Sign algorithm is a PPT algorithm that, given the system public parameters* pms*, a secret key sk_{id} and a message m, outputs a signature σ.*
- *The Verify algorithm is a polynomial-time algorithm that takes as inputs the public parameters* pms*, a public key pk_{id}, a message m and a signature σ, and outputs 1 if σ is a valid signature on the message m with secret key sk_{id}; it outputs 0 otherwise.*

2.4 Partially blind signature schemes

Blind signature protocols are interactive protocols between a user, a signer and a verifier, in which the signer produces a digital signature of a message submitted by the user, but does not learn anything about the contents of the message. This primitive was introduced by Chaum in [31] and has since been used in a vast array of privacy-related protocols, such as e-cash, electronic voting and anonymous credential systems. An inherent drawback of blind signature protocols is that the signer cannot enforce a certain format on the message. Traditionally, this problem has been solved using cut-and-choose techniques, in which the requester of a signature generates and blinds a number n of messages, the signer asks the requester to unblind all messages but a randomly chosen one, checks whether all unblinded messages conform to the required format and, if yes, signs the only message that remains blinded. Using cut-and-choose techniques solves the problem (the probability that the requester succeeds in getting a non-conforming message signed is upper-bounded by $1/n$), but it does so at the cost of high computation and communication overheads.

Partially blind signatures were introduced by Abe in [6, 7] as an alternative to cut-and-choose protocols. In a partially blind signature protocol, the user and the signer agree on a public information that is to be included in the signed message. Both the user and the signer can be sure that such an information is really included in the signature, and the secret message of the user remains blinded to the signer.

Definition 7 (Partially blind signature scheme [98])**.** *A partially blind signature consists of three participants: signer, user and verifier. There are three algorithms: Key Generation, Partially Blind Signature Issuance, and Verification.*

- *Key Generation is a probabilistic polynomial-time algorithm that takes as input a security parameter γ and outputs a key pair (pk, sk).*

- *Partially Blind Signature Issuance is an interactive protocol between the signer and the user. The public inputs of both the user and signer contain the previously agreed upon public information* info*. The private input of the signer is sk, and the private input of the user is the message m to be signed. Upon protocol completion, the private output of the user contains either* `fail` *or* $(\mathsf{info}, m, \sigma)$*, where σ is the signer's signature on* info *and m.*

- *Verification is a polynomial-time algorithm that takes* $(pk, \mathsf{info}, m, \sigma)$ *as input and outputs either* `accept` *or* `reject`.

Definition 8 (Correctness and security of a partially blind signature scheme)**.** *A partially blind signature scheme is considered correct and secure if it satisfies the following properties:*

- *(Correctness) For honestly generated parameters, and for honest execution of Partial Blind Signature Issuance, the resulting partial blind signature passes verification with overwhelming probability.*
- *(Partial blindness) The following must hold: 1) the signer must be assured that the embedded public information is included in the signature, and that no one can modify this information; and 2) based on the embedded information, a signer cannot link the signature with the concrete issuing instance that produced it.*
- *(Unforgeability) A partially blind signature scheme is called unforgeable against one-more forgery under chosen-message attack if, for some integer* ℓ*, and a given public information* info*, there is no probabilistic polynomial-time adversary* $\mathcal{A}$ *that can compute, after* ℓ *interactions with the signer,* $\{(\mathsf{info}, m_j, \sigma_j)\}_{j=1,\ldots,\ell+1}$ *valid signatures with non-negligible probability.*

Boldyreva proposed in [19] a blind version of the BLS signature scheme from [26]. The security of Boldyreva's scheme rests on the chosen-target version of the Computational Diffie-Hellman problem (see Definition 3) in Gap Diffie-Hellman groups. In [97], Zhang *et al.* proposed a signature scheme based on the $k+1$ Exponent Problem. Later, the same authors published in [98] a blind version of the previous scheme, following the same approach of Boldyreva.

2.5 Zero-knowledge proofs

A zero-knowledge proof (ZKP), as introduced by Goldwasser *et al.* in [52], is a method whereby a party (the prover) can prove to another party (the verifier) that a given statement is true, without leaking any information from this process beyond the fact that the statement is true. ZKPs are typically used as building blocks for other cryptographic protocols as protection against cheaters or malicious participants. For example, in multiparty computation protocols, ZKPs can be used to ensure that the inputs given by the participants are of a given form (that is, participants follow the protocol correctly).

The Schnorr [85] ZKP protocol, for example, is used to prove that $x \in \mathbb{Z}_p$ is the discrete logarithm of a public value $y = g^x$ in the cyclic group $\mathbb{G}$ with generator g. This ZKP can be used as an identification protocol, when x is a secret identifier of a user (Schnorr identification scheme).

The Fiat-Shamir [48] heuristic can be used to turn interactive ZKPs into non-interactive ZKPs (NIKZs). The non-interactive version of the Schnorr identification scheme is known as the Schnorr signature scheme. This procedure has been used to build other digital signature schemes from interactive zero knowledge proofs. The Groth-Sahai [55] proof framework is used to build non-interactive zero knowledge proofs of statements formulated in terms of any operation associated with bilinear groups.

In [70], the authors introduce a general framework based on one-way homomorphisms to prove in zero-knowledge the knowledge of the preimage of a group homomorphism. They show that the Schnorr protocol [85] is an instance of this general framework, using the exponentiation in groups in which the discrete logarithm problem is hard. Our loyalty programs construction uses a variation of this protocol, based on the exponentiation in bilinear groups in which the discrete logarithm problem is assumed hard.

2.6 Secure multiparty computation for set intersection

Secure multiparty computation (MPC) allows a set of parties to compute functions of their inputs in a secure way without requiring a trusted third party. During the execution of the protocol, the parties do not learn anything about each other's input except what is implied by the output itself. There are two main adversarial models: honest-but-curious adversaries and malicious adversaries. In the former model, the parties follow the protocol instructions but they try to obtain information about the inputs of other parties from the messages they receive. In the latter model, the adversary may deviate from the protocol in an arbitrary way. Aumann and Lindell [10] introduced a new model, the covert adversaries. A covert adversary may deviate from the protocol in an attempt to cheat, but such deviations are detected by honest parties. In this context, the parties may be considered rational, that is, acting according to their interests. In game-theoretic terms, it is assumed that players only try to maximize their utility functions; hence, all possible deviations from the correct protocol execution have this goal.

We will restrict here to a two-party setting in which the input of each party is a set, and the desired output is the cardinality of the intersection of both sets. The intersection of two sets can be obtained by using generic constructions based on Yao's garbled circuit [96]. This technique allows computing any arithmetic function, but for most of the functions it is inefficient. Many of the recent works on two-party computation are focused on improving the efficiency of these protocols for particular families of functions.

Freedman, Nissim, and Pinkas [49] presented a more efficient method to compute the set intersection, a *private matching scheme*, that is secure in the honest-but-curious model. A private matching scheme is a protocol between a customer $\mathcal{C}$ and a server $\mathcal{S}$ in which $\mathcal{C}$'s input is a set X of size $i_{\mathcal{C}}$, $\mathcal{S}$'s input is a set Y of size $i_{\mathcal{S}}$, and at the end of the protocol $\mathcal{C}$ learns $X \cap Y$. The scheme uses polynomial-based techniques and homomorphic encryption schemes. Next we present an outline of the scheme in [49] that is secure in the honest-but-curious model:

1. $\mathcal{C}$ computes the polynomial $p(x) = \prod_{i=1}^{s}(x - a_i)$.

2. $\mathcal{C}$ sends $Enc(p_0), \ldots, Enc(p_s)$ to $\mathcal{S}$, where p_i is the coefficient of degree i of p.

3. For every $1 \leq j \leq t$, $\mathcal{S}$ picks a random element $r_j \in \mathbb{Z}_n$ and computes $Enc(r_j \cdot p(b_j) + b_j)$. Observe that these ciphertexts can be obtained from $Enc(p_0), \ldots, Enc(p_s)$ and $Enc(b_j)$ by using the homomorphic properties of the cryptosystem. Then $\mathcal{S}$ sends the ciphertexts to $\mathcal{C}$.

4. $\mathcal{C}$ decrypts the t ciphertexts. The result of each decryption is an element from $X \cap Y$ or a random element.

If the size of the domain of Enc is much larger than $|X|$ the scheme computes $X \cap Y$ with high probability.

Several variations of the private matching scheme were also presented in [49]: an extension to the malicious adversary model, an extension of the multi-party case, and schemes to compute the cardinality of the set intersection and other functions. Constructing efficient schemes for set operations is an important topic in MPC and has been studied in many other contributions. Several works such as [16, 37, 57, 66, 92] present new protocols to compute the set intersection cardinality.

2.7 Paillier's cryptosystem

In this cryptosystem, the public key consists of an integer n (product of two RSA primes), and an integer g of order n modulo n^2, for example, $g = 1 + n$. The secret key is $\phi(n)$, where $\phi(\cdot)$ is Euler's totient function.

Encryption of a plaintext integer m, with $m < n$ involves selecting a random integer $r < n$ and computing the ciphertext c as

$$c = Enc(m) = g^m \cdot r^n \bmod n^2 = (1 + mn)r^n \bmod n^2.$$

Decryption consists of first computing $c_1 = c^{\phi(n)} \bmod n^2 = 1 + m\phi(n)n \bmod n^2$ and then $m = (c_1 - 1)\phi(n)^{-1} \bmod n^2$.

The homomorphic properties of Paillier's cryptosystem are as follows:

- *Homomorphic addition of plaintexts.* The product of two ciphertexts decrypts as the sum of their corresponding plaintexts:

  $$D(E(m_1, r_1) \cdot E(m_2, r_2) \bmod n^2) = m_1 + m_2 \bmod n.$$

  Also, the product of a ciphertext times g raised to a plaintext decrypts as the sum of the corresponding plaintexts:

  $$D(E(m_1, r_1) \cdot g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n.$$

- *Homomorphic multiplication of plaintexts.* An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts:

  $$D(E(m_1, r_1)^{m_2} \bmod n^2) = D(E(m_2, r_2)^{m_1} \bmod n^2) = m_1 m_2 \bmod n.$$

  More generally, given a constant k, $D(E(m_1, r_1)^k \bmod n^2) = km_1 \bmod n$.

2.8 Bloom filters

Bloom filters, as proposed by Bloom [17], are probabilistic space-efficient data structures that encode datasets while still allowing membership queries. The latter queries have a false positive rate proportional to the number of encoded elements. A Bloom filter consists of a bit array $B = b_0, \ldots, b_{m-1}$ of length m, with all bits initially set to 0, and is equipped with k different hash functions with range $[0, \ldots, m-1]$ each of which maps some set element to one of the m array

positions with a uniformly random distribution. Typically $k \ll m$. An element e to be inserted is hashed with the k hash functions, and the corresponding bits $b_{h_0(e)}, \ldots, b_{h_{k-1}(e)}$ are set to 1. Accordingly, a membership query for element e is performed by checking whether $b_{h_0(e)} = \cdots = b_{h_{k-1}(e)} = 1$.

If a Bloom filter contains n elements, the probability of false positive (that is, the probability of the membership query answering that an element not in the set belongs to it) is given by

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k, \tag{2.1}$$

that is the probability that the indices obtained by hashing the element with the k hash functions are all set to 1, either by another single element (which is unlikely, because it would mean that the k hash values of the second element collide with those of the first one) or by a combination of other inserted elements. Note that Expression (2.1) is only an approximation, because it assumes that a bit is set to 1 independently of the values of the other bits, which is a simplification of reality [27].

The number of elements encoded in a Bloom filter (*i.e.* the cardinality of the encoded set S) is approximated as

$$|S| \approx -\frac{m}{k} \ln\left(1 - \frac{H(B_S)}{m}\right), \tag{2.2}$$

where B_S is the bit array resulting from encoding S, $H(\cdot)$ is the Hamming weight of a bit array (number of bits set to 1), m is the length of the Bloom filter and k is the number of hashes [88].

2.8.1 Union and intersection of sets

The union and intersection of encoded sets can be easily computed by performing the bitwise $\vee$ (or) and $\wedge$ (and) operations, respectively. If we assume the sets A and B and their respective encodings B_A and B_B are of the same size m, with B_A and B_B having both k hash functions, the Bloom filter

$$B_{A \cap B} = b_{A,0} \wedge b_{B,0}, \ldots, b_{A,m-1} \wedge b_{B,m-1}$$

represents the encoding of the intersection set $A \cap B$. Likewise, the union set can be obtained by computing the bitwise $\vee$ operation.

Note that a Bloom filter obtained by encoding $A \cap B$ or $A \cup B$ will not be exactly equal to the Bloom filters obtained by performing the above operations. This is due to the underlying probabilistic nature of Bloom filters, and may cause some elements to be lost or the false positive rate to increase.

Finally, by applying Expression (2.2) to the resulting Bloom filters, we can obtain estimates of the cardinalities of the union and the intersection of sets. We plan to use the cardinality of the intersection of two sets to compute their distance.

2.8.2 Security and privacy of Bloom filters

Gerbet *et al.* [51] give security definitions for Bloom filters, by deriving them from the standard security properties of cryptographic hash functions.

Definition 9 (Support)**.** *The support of a vector B of size m, denoted as $\mathsf{supp}(B)$, is the set of its non-zero coordinate indices:*

$$\mathsf{supp}(B) = \{i \in [0, \ldots, m-1], b_i \neq 0\}.$$

Definition 10 (Pre-image of a Bloom filter)**.** *Given a Bloom filter B, a pre-image of the filter is a string $y \in \{0,1\}^*$ with indices $I_y = \{h_0(y), \ldots, h_{k-1}(y)\} \subseteq \mathsf{supp}(B_x)$.*

In a *pre-image attack*, an adversary is given B. Finding pre-images becomes easier as the support of the filter increases (as a consequence of the insertion of more elements). In the extreme case in which the size of the support equals the length m of the filter, finding a pre-image is trivial. This is why *we define a pre-image attack to be successful only if the attacker finds a pre-image y that coincides with an element inserted in B.* Even if the attacker does not know the elements inserted in B, he may be able to recognize one when he hits it, due to any redundancy in the element (for example, an URL is easy to recognize).

Definition 11 (Second pre-image of a Bloom filter)**.** *Given a Bloom filter B_x containing an element $x \in \{0,1\}^*$ with indices $I_x = \{h_0(x), \ldots, h_{k-1}(x)\}$, a second pre-image of the filter is another string $y \neq x$ with $I_y = \{h_0(y), \ldots, h_{k-1}(y)\}$ such that $I_y \subseteq \mathsf{supp}(B)$.*

In a *second pre-image attack*, an adversary is given B_x and its element x. The attack succeeds if the attacker finds a second pre-image $y \neq x$ that also passes the membership test.

Definition 12 (Collision in a Bloom filter)**.** *Given a Bloom filter B, two strings $x \in \{0,1\}^*$ and $y \in \{0,1\}^*$, with $x \neq y$, $I_x = \{h_0(x), \ldots, h_{k-1}(x)\}$, and $I_y = \{h_0(y), \ldots, h_{k-1}(y)\}$ are a collision if both $I_x \subseteq \mathsf{supp}(B)$ and $I_y \subseteq \mathsf{supp}(B)$.*

In a *collision attack*, an adversary is given B. The attack succeeds if the adversary finds a collision.

In the system we propose, servers store the users' profiles encoded as Bloom filters. Therefore:

- In order to preserve the privacy of the users, it is important that the server, or any attacker with access to the server, cannot obtain the profiles of the users from their encoded versions. Thus, we need Bloom filters to resist pre-image attacks in order to ensure the privacy of the users.
- On the other hand, resistance to second pre-image attacks is needed to ensure that the server or any attacker cannot replace a known legitimate profile by a fabricated profile that also passes the authentication test.
- Finally, resistance to collision attacks is needed to ensure that it is not easy for the attacker to find a random profile that passes the authentication test as if it was some legitimate profile (unknown to the attacker).

Gerbet *et al.* [51] also provide the computational complexities of pre-image and second pre-image attacks against Bloom filters, and point out that typical implementations of Bloom filters use non-cryptographic hash functions for efficiency reasons and are thus weak against these attacks. They recommend using keyed hash functions, such as message authentication codes (*e.g.* HMAC) to thwart attacks against Bloom filters. Using HMAC does not only increase the domain of the hash function (the HMAC input is augmented with the key), but also makes the computation slower, which is a desirable property to keep hash functions secure against brute-force search.

2.8.3 Secure instantiation of the Bloom filters

We follow the construction in Kirsch and Mitzenmacher [65] to build our Bloom filters. Basically, the k hash functions $h_i(x)$ are constructed from two independent hash functions $g_1(x)$ and $g_2(x)$ as $h_i(x) = g_1(x) + ig_2(x) \mod m$. This strategy reduces the computing time to encode the sets, while maintaining the false positive rate as low as if k independent hash functions were used. In addition, to ensure that the domain of the $h_i(x)$ is large, we take $g_2(x)$ to be keyed, that is, $g_2(x, \mathsf{key})$, so that $h_i(x)$ is also keyed, that is, $h_i(x, \mathsf{key})$.

The optimal values for m and k can be computed as a function of the maximum number N of allowed element insertions, and the maximum acceptable false positive rate ρ [51]:

$$m = -\frac{N \ln \rho}{(\ln 2)^2}, \qquad k = \frac{m}{N} \ln 2. \tag{2.3}$$

2.8.4 Bloom filters in privacy-preserving data mining

In Dong *et al.* [44], the authors propose a private matching scheme based on garbled Bloom filters and oblivious transfer that overcomes the scalability problem of previous proposals. In this protocol, the datasets are encoded as garbled Bloom filters (an extension of Bloom filters). The parties then engage in an oblivious transfer protocol to compute a new Bloom filter that encodes the intersection of the original sets. Pinkas *et al.* [77] improve on the efficiency of the previous protocol. The mechanism described in Kerschbaum [64] uses the homomorphic properties of the Goldwasser-Micali cryptosystem to test membership of elements in encrypted Bloom filters (the Bloom filter is encrypted bit by bit). Schnell *et al.* [84] also use Bloom filters to link records from vertically partitioned data with encrypted identifiers.

Bloom filters have been used in other applications of privacy-preserving data mining; for example, to implement a secure dot product protocol to derive association rules from vertically partitioned data [62].

2.9 Anonymous payment mechanisms

Throughout this work, and specifically when dealing with group and loyalty discount protocols, we will reach stages in which users have to pay. It would certainly be pointless to use any kind of non-anonymous payment mechanism, such as a credit card or PayPal, after engaging in protocols that strive to preserve the anonymity of users.

Therefore, we recall available payment methods that preserve the payer's anonymity. We do not intend to offer an exhaustive list, but merely to show that there are several options.

The simplest option for an anonymous payment method is to use cash if the application and the service provider allow it. Unfortunately, this will not always be the case, and other payment methods have to be taken into account. Electronic cash protocols such as [32] are good candidates for this role. Nowadays, Bitcoin [72] is a well-established electronic currency and, although it is

not anonymous by design [81], it can be a good solution if accompanied by careful key management policies. Also, extensions of the original protocol such as Zerocoin [71] provide anonymity by design.

For completeness, we propose in this work to resort to a much simpler approach, based on prepaid scratch cards that users can buy at certain points of sale using cash (for maximum anonymity). Each such card contains a code Pay.Code which the card provider will associate with a temporary account holding a fixed amount specified by the card denomination. A well-known example of this type of prepayment system that can be used to buy on-line services and products at a variety of vendors worldwide is Paysafecard [76].

2.10 Short-range communication technologies

Our group accreditation mechanism requires communication among the members (their devices) of a group. Additionally, at least one of the group members has to be able to communicate with the automated verifier. The choice of the communication technologies heavily depends on the kind of service our accreditation mechanism is used for. For example, deploying our method in an on-line store has different communication requirements (and probably different functional requirements) than deploying it in a toll station. Moreover, if the verifier wants to learn not only the number of group members, but also if they are physically together, the choice of communication technologies becomes more restricted.

In the on-line setting, we propose to use anonymous communication channels, such as the Tor network, to communicate with the verifying entity. Since physical closeness is not very relevant in the on-line world, our accreditation method needs only to be used to prove the size of the group.

The full potential of our mechanism can be leveraged in a physical setting, such as a toll station in high-occupancy lanes, where, beyond verifying the group size with our method, the choice of communication technologies can help verifying that the group members are physically together. For these cases, short-range communication technologies, such as NFC, Bluetooth or WiFi, are suitable. It is desirable that communication establishment be fast and not too cumbersome to the user.

We propose using Bluetooth, and in particular Bluetooth Low Energy (BLE, [18]) to communicate with the verifying device. BLE solves some of the main limitations of traditional Bluetooth, *i.e.* it reduces detection and bonding times, requires much less work by the user than NFC and has a shorter range than

both Bluetooth and WiFi, which is desirable in a method like ours. Specifically, while a Bluetooth connection can reach as far as 100 m if using Class 1 chips, smartphones carry Class 2 Bluetooth chips, which provide ranges of about 10 m. Furthermore, Bluetooth is heavily affected by physical obstacles and the effective ranges will typically be less than 10 m. Such a range is appropriate for smartphones to communicate with the verifying device without suffering interference from any smartphone that is not in the very close vicinity (*e.g.*, that is not in the car in the case of toll discounts). As far as availability is concerned, BLE is implemented by most major smartphone manufacturers.

Regarding communication between the smartphones of group members, we propose to rely on NFC. Given the effective range of NFC (about 2 to 5 cm in smartphones), group members cannot collaborate in the protocol unless they are very close to each other (*i.e.* they have to sit in the same car in the case of toll discounts).

Chapter 3

Group Size Accreditation Method

3.1 Introduction

Group discounts are offered by vendors and also public authorities to encourage consumers (or citizens) to cluster in groups when using services. Dealing with groups of consumers (rather than with a single consumer at a time) can lead to a more efficient use of the available resources, less impact on the environment or other specific benefits. Two representative examples of group discounts are the following: 1) group tickets for museums, stadiums or leisure parks, which may allow service providers to plan activities more efficiently ahead of time; 2) discounted highway tolls or parking fees for high-occupancy vehicles (HOVs), which aim to reduce traffic congestion and pollution.

Although it is common and simplest for vendors to require all group members to identify themselves, we observe that the privacy loss this inflicts on consumers is not really justified in most applications. We make the assumption that normally the only relevant feature about a group is the *number of its members*, rather than their identities or other features. Other features that might sometimes be of interest are the age of the participants, their location or whether they are physically together or not.

Certifying that a group of people is of a minimum size, along with other features of the group, such as its members being physically close to each other, is trivial in a face-to-face setting with a human verifier who can see that the

required minimum amount of people is present (although even in this case the human verifier could be tricked by colluding groups). However, checking the size of a group becomes far from obvious for an automatic verifier or in an on-line setting, especially when considering the anonymity of group members.

3.1.1 Contributions

In this Chapter, we describe a method to certify the number of members in a group formed on the fly, while preserving the anonymity of the members and with no specific dedicated hardware requirements. Namely, our mechanism only requires every group member to have a computing device with some communication capabilities, *e.g.* a smartphone. Also, we explore the option to include payment in our proposed system, which is necessary for group discounts. We complete the description of our method with a possible anonymous payment mechanism, based on scratch cards.

Our group size accreditation method is based on an identity-based dynamic threshold ($IBDT$) signature scheme, namely a variant of the second protocol proposed in [56], but adapted to the asymmetric pairing setting. The protocol suite introduced here is a generalization of a specific protocol for high-occupancy vehicle toll discounts for which we filed patent [42].

The contributions in this Chapter have been published in [41]. An extended version has been submitted to a journal.

The Chapter is structured as follows. Section 3.2 briefly recalls past works related to group size accreditation mechanisms. Section 3.3 describes the new IBDT primitive; its security requirements are identified and a proof of its security is given. Section 3.4 introduces the key management scheme used in our proposal. Section 3.5 describes our group size accreditation method, including the required entities and protocols. The security and the privacy of our method are analyzed in Section 3.6. In Section 3.7, we give a complexity estimation of our approach and describe precomputation optimizations. Section 3.8 presents simulation results. Section 3.9 describes a use case implementation of our system, focused on parking tolls for high-occupancy vehicles. Finally, Section 3.10 summarizes conclusions of this Chapter.

3.2 Related work on group size accreditation methods

A group size accreditation method is a system by which the members of a group of users can be counted reliably, usually as a requisite to grant them group discounts or other special service conditions. Such a system must be robust against cheaters, because cheating endangers the benefits of group discounts outlined in the previous section. The traditional solution is to have an employee count the number of members in each group. This approach may be good in some cases, such as small events, but when lots of people participate and/or a short response time is needed, such as in HOV lanes, automated mechanisms are essential.

Automated mechanisms may involve using cameras, detecting the users' mobile devices, etc. Unfortunately, most of these systems are not privacy-aware: beyond counting members, they allow identifying them. Worse yet, some technologies that invade privacy, such as camera-based ones, can still fail to identify cheaters [30]. A mechanism that is highly effective against cheaters is described in [35], in which dedicated devices installed in cars count and recognize drivers and passengers by measuring the strength of the Bluetooth or WiFi signal of their mobile devices. In this case, counting the occupants of vehicles is done to grant access to HOV lanes. This mechanism, though, requires installing special hardware in cars and identifies the occupants of the vehicle. So, while it does prevent cheating, it still poses a privacy risk to the users.

To the best of our knowledge, no mechanisms have been proposed that effectively and unequivocally ascertain the number of members of a group while preserving member anonymity and not requiring specific hardware.

3.3 Identity-based dynamic threshold signatures

We present here our new cryptographic primitive that combines dynamic threshold signatures and identity-based signatures. An *identity-based dynamic threshold signature* $\mathsf{IBDTS} = (\mathsf{IBDT.Setup}, \mathsf{IBDT.Keygen}, \mathsf{IBDT.Sign}, \mathsf{IBDT.Comb}, \mathsf{IBDT.Verify})$ consists of five probabilistic polynomial-time (PPT) algorithms:

- $\mathsf{IBDT.Setup}(1^\lambda, \mathcal{ID}, n)$ is the randomized *trusted set-up* algorithm taking as input a security parameter λ, a universe of identities $\mathcal{ID}$ and an integer $n \in \mathsf{poly}(\lambda)$ which is an upper bound on the size of the threshold policies. It outputs a set of public parameters pms (which contains λ, $\mathcal{ID}$ and n),

as well as a master secret key msk and the corresponding master public key mpk. An execution of this algorithm is denoted as $(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBDT.Setup}(1^\lambda, \mathcal{ID}, n)$.

- $\mathsf{IBDT.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \mathbf{id})$ is a *key extraction* algorithm that takes as inputs the public parameters pms, the master keys mpk and msk, and an identity $\mathbf{id} \subset \mathcal{ID}$. Since in what follows we use only one identity-based public key per user, we can assimilate this key to the user's identity and denote it as $\mathbf{id}$; from Section 3.3.5 onwards, we will attribute more than one identity-based public key to each user and we will need to denote the identity-based public keys in a different way. The output of the key generation algorithm is a secret key $SK_{\mathbf{id}}$. We write $SK_{\mathbf{id}} \leftarrow \mathsf{IBDT.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \mathbf{id})$ to denote an execution of this algorithm.

- $\mathsf{IBDT.Sign}(\mathsf{pms}, \mathsf{mpk}, SK_{\mathbf{id}}, \mathsf{Msg}, \Gamma)$ is a randomized *signing* algorithm which takes as input the public parameters pms, the master public key mpk, a secret key $SK_{\mathbf{id}}$, a message Msg and a threshold signing policy $\Gamma = (t, S)$ where $S \subset \mathcal{ID}$ and $1 \leq t \leq |S| \leq n$. It outputs a partial signature $\sigma_{\mathbf{id}}$. We denote an execution of this algorithm as $\sigma_{\mathbf{id}} \leftarrow \mathsf{IBDT.Sign}(\mathsf{pms}, \mathsf{mpk}, SK_{\mathbf{id}}, \mathsf{Msg}, \Gamma)$.

- $\mathsf{IBDT.Comb}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \Gamma, \{\sigma_{\mathbf{id}}\}_{\mathbf{id} \in S_t})$ is a deterministic *signing* algorithm which takes as input the public parameters pms, the master public key mpk, a message Msg, a threshold signing policy $\Gamma = (t, S)$ and the partial signatures of some set $S_t \subset S$, $|S_t| \geq t$ and computes a global signature σ. It outputs a signature σ. We denote the action taken by the signing algorithm as $\sigma \leftarrow \mathsf{IBDT.Comb}(\mathsf{pms}, \mathsf{mpk}, SK_{\mathbf{id}}, \mathsf{Msg}, \Gamma, \{\sigma_{\mathbf{id}}\}_{\mathbf{id} \in S_t})$.

- $\mathsf{IBDT.Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \sigma, \Gamma)$ is a deterministic *verification* algorithm taking as input the public parameters pms, a master public key mpk, a message Msg, a signature σ and a threshold predicate $\Gamma = (t, S)$. It outputs 1 if the signature is deemed valid and 0 otherwise. We write $b \leftarrow \mathsf{IBDT.Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \sigma, \Gamma)$ to refer to an execution of the verification protocol.

For correctness, for any $\lambda \in \mathbb{N}$, any integer $n \in \mathsf{poly}(\lambda)$, any universe $\mathcal{ID}$, any set of public parameters and master key pair $(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBDT.Setup}(1^\lambda, \mathcal{ID}, n)$, and any threshold policy $\Gamma = (t, S)$ where $1 \leq t \leq |S|$, it is required that

$$\sigma \leftarrow \mathsf{IBDT.Comb}\big(\mathsf{pms}, \mathsf{mpk}, SK_{\mathbf{id}}, \mathsf{Msg}, \Gamma, \{\sigma_{\mathbf{id}}\}_{\mathbf{id} \in S_t}\big)$$

$$\mathsf{IBDT.Verify}\big(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \sigma\big) = 1$$

whenever: a) the values pms, mpk and msk have been obtained by properly executing the algorithms IBDT.Setup, b) $|S_t| \geq t$, and c) for each $\mathbf{id} \in S_t$, $\sigma_{\mathbf{id}} \leftarrow \mathsf{IBDT.Sign}\big(\mathsf{pms}, \mathsf{mpk}, SK_{\mathbf{id}}, \mathsf{Msg}, \Gamma\big)$ and $SK_{\mathbf{id}} \leftarrow \mathsf{IBDT.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \mathbf{id})$.

3.3.1 Related cryptographic techniques

We address the problem of a *dynamic* group of users (formed on the fly) *proving its size* t in an *anonymous* way, where t is some publicly declared value. In our system, users have unique identifiers (*e.g.* their national ID card numbers). When a group of users accredits its size using our protocol, the only information that is revealed about the group members is the set of the j-th digits of their unique identifiers, which are all different, for some j, as well as the value j. Assuming the unique identifiers are ℓ decimal digits long, there are up to $10^{\ell-1}$ users sharing a certain value of the j-th digit, so anonymity is well preserved. More specifically, the probability to identify a group of t users given that only the j-th digits of their identifiers are known and all these j-th digits are different is $10^{-(\ell-1)t}$. The reason is that, when the j-th digits of the t group members are fixed, known and all different, there are $10^{(\ell-1)t}$ possible assignments for the remaining $(\ell-1) \times t$ digits of the t identifiers, and each assignment yields a different set of t identifiers (there are no repeated identifiers, since the j-th digits are all different).

To achieve the goals in the previous paragraph, we make a special use of an IBDT signature scheme (see Section 3.3.3 for a description of it and Section 3.4 on how we use it in our protocol). There are other cryptographic techniques offering some of the three desirable features mentioned above (member counting, anonymity and dynamicity), which we discuss below.

- Our same IBDT signature scheme could be used in a different way, to directly prove that t out of n possible signers have collaborated to sign a document, where the group of n possible signers is dynamically chosen. The shortcoming is that IBDT signatures reveal the set of n signers who can collaborate to produce a signature. Therefore, the anonymity level is at best $\binom{n}{t}^{-1}$ (this best case occurs when the subgroup of t signers is not leaked, which is an additional security feature that is not satisfied by every IBDT signature scheme).

- Threshold ring signatures also allow making sure that at least t users out of n have collaborated to compute a signature. However, the public

keys of the users are not identities, which makes key management more complicated. Identity-based threshold ring signatures would be a better solution, but again the anonymity level achieved would be at most $\binom{n}{t}^{-1}$.

- Zero-knowledge proofs could be used to prove knowledge of t-out-of-n secret keys out of a group of n keys. There are several alternatives to prove this under different assumptions, but to the best of our knowledge, they are all linear in n. Indeed, one alternative is to prove this using Groth-Sahai NIZK proofs [55] in bilinear groups and improvements thereof which specifically try to improve GS proofs for this type of "threshold statement" [80, 53]. In this case, all the proofs are linear in n. On the other hand, we note that the recent extremely efficient quasi-adaptive NIZK proofs in bilinear groups ([60], and improvements thereof [61, 68]) allow only proving membership in linear spaces of a discrete-logarithm group $\mathbb{G}$ or are designated-verifier [5], and we do not know how to use them to prove this type of threshold statement (or more generally, to prove "knowledge" of a witness). Further, approaches in the random oracle model, like the sigma protocol to prove threshold statements of [34], also result in a linear proof. Finally, one could use succint non-interactive arguments of knowledge (SNARKs) to prove such statement in constant size, but these are based on very strong and controversial hardness assumptions (knowledge-of-exponent type of assumptions) which we prefer to avoid. In summary, with zero-knowledge proofs, the proof size would be linear in n and the anonymity level would be also at best $\binom{n}{t}^{-1}$.

- Distributed group signatures do offer anonymity. However, the groups in such signatures cannot be created on the fly (rather, each group has a manager). Note that although in the literature of group signatures one can find solutions in what is called a "dynamic group setting" [11, 67], the groups are always controlled by a manager who distributes the keys to group members. In this context, the term dynamic refers to the fact that users may join or leave the group after setup, while we want groups to be created on the fly by the users themselves.

Hence, direct application of any of the above cryptographic techniques provides at best an anonymity level $\binom{n}{t}^{-1}$. For this level to match the one offered by our scheme $\binom{n}{t}$ should be as large as $10^{(\ell-1)t}$, which demands a large n. While choosing a large n is good for privacy, it takes a heavy toll on the efficiency of any of these solutions. Indeed, in the public-key setting, the verifier must have

access to at least the n public keys that define the ring, plus the corresponding certificates. On the other hand, identity-based solutions avoid the public key and certificate management problem but, in all of their instantiations in the literature, the public parameters define some upper bound on n, called hereafter n', so that a) either the size of the signature (or the size of the zero-knowledge proof) is at least linear in n' —which makes verification slow—, or b) the size of the secret key is at least linear in n'[1].

Although we run into a similar problem when using as a building block an IBDT signature scheme (our IBDT signature scheme has a constant-size signature but the size of the secret keys depends on n'), by using a more complex key management, we can set n' to be equal to the largest group size t' that makes sense in the specific application under consideration. In most applications, the largest possible group size t' is much smaller than the total number of possible signers (in the vehicular application, t' is the maximum number of people that can travel in a vehicle).

The reason why in our solution we can choose n' to be independent of the privacy level is that we obtain anonymity not from the cryptographic primitive itself but by defining the protocol in such a way that it only reveals one digit of the signers' identity, as sketched in the first paragraph of this section.

Finally, we justify the use of an IBDT signature instead of t copies of a normal identity-based signature, each separately computed by a different user. The downside of the latter option is that it is more costly for the verifier. Indeed, each signature verification involves computing pairings, which are very expensive operations. Hence, with t separate signatures, the number of pairings to be computed is linear in t, whereas it is constant when verifying one IBDT signature.

3.3.2 Security model of IBDTs

An IBDT signature scheme must satisfy the usual property of *unforgeability*. We consider a relaxed notion where the attacker *selects* the signing policy $\Gamma^\star = (t^\star, S^\star)$ that he wants to attack at the beginning of the game (here $\Gamma^\star$ means that a signature is valid if jointly produced by at least $t^\star$ signers from a set $S^\star$ of possible signers). However, the message $\mathsf{Msg}^\star$ whose signature is eventually forged is not selected in advance. The attacker can ask for valid signatures for messages and signing policies of his adaptive choice. The resulting property

[1] Even if there are ring signatures of constant size [40], we are not aware of constant-size identity-based threshold ring signatures.

of *selective-predicate and adaptive-message unforgeability under chosen-message attacks* (sP-UF-CMA, for short) is defined in terms of the following game.

Definition 13. *Let λ be an integer. Consider the following game between a probabilistic polynomial-time (PPT) adversary $\mathcal{F}$ and its challenger.*

Initialization. *The challenger begins by specifying a universe of identities $\mathcal{ID}$ as well as an integer $n \in \mathsf{poly}(\lambda)$, which are sent to $\mathcal{F}$. Then, $\mathcal{F}$ selects a subset $S^\star \subset \mathcal{ID}$ of signers such that $|S^\star| \leq n$ and a threshold $t^\star \in \{1, \ldots, |S^\star|\}$. These define a threshold predicate $\Gamma^\star = (t^\star, S^\star)$.*

Setup. *The challenger runs $(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}) \leftarrow \mathit{IBDT.Setup}(1^\lambda, \mathcal{ID}, n)$ and sends $\mathsf{pms}, \mathsf{mpk}$ to the forger $\mathcal{F}$.*

Queries. *$\mathcal{F}$ can interleave secret key and signature queries.*

- ***Secret key queries.*** *$\mathcal{F}$ can adaptively request the secret keys of any identity $\mathbf{id}$ under the restriction that the total number of queried identities in the set $S^\star$ is strictly less than $t^\star$. As an answer to such a query, the adversary receives $SK_{\mathbf{id}} \leftarrow \mathit{IBDT.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \mathbf{id})$.*
- ***Signature queries.*** *$\mathcal{F}$ adaptively chooses a pair (Msg, Γ) consisting of a message Msg and a threshold predicate $\Gamma = (t, S)$ such that $1 \leq t \leq |S| \leq n$. The challenger replies with a valid signature for Msg and the policy Γ.*

Forgery. *At the end of the game, $\mathcal{F}$ outputs a pair $(\mathsf{Msg}^\star, \sigma^\star)$. We say that $\mathcal{F}$ is successful if:*

- *$\mathit{IBDT.Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}^\star, \sigma^\star, \Gamma^\star) = 1$, and*
- *$\mathcal{F}$ has not made any signature query for the pair $(\mathsf{Msg}^\star, \Gamma^\star)$.*

The forger's advantage in breaking the sP-UF-CMA security is defined as

$$\mathsf{Succ}^{\mathit{sP\text{-}UF\text{-}CMA}}_{\mathcal{F},\mathit{IBDT}}(\lambda) = \Pr[\mathcal{F} \textit{ wins}].$$

An identity-based dynamic threshold signature IBDTS is selective-predicate adaptive-message unforgeable *(or* sP-UF-CMA unforgeable*) if, for any PPT adversary $\mathcal{F}$, $\mathsf{Succ}^{\mathit{sP\text{-}UF\text{-}CMA}}_{\mathcal{F},\mathit{IBDT}}(\lambda)$ is a negligible function of λ.*

3.3.3 An instance of an IBDT signature scheme

We instantiate an IBDT signature scheme by adapting the threshold attribute-based signature scheme of [56], which builds on the attribute-based encryption scheme of [9]. If one identifies attributes with identities, threshold attribute-based signature schemes are closely related to identity-based dynamic threshold

signature schemes, except that the former have an additional property called *collusion resistance*. Collusion resistance means that two users who individually do not satisfy a signing policy (t, S) but such that the sum of their attributes does, cannot combine their secret keys to sign a message for the policy (t, S).

For identity-based threshold signature schemes, we require precisely the opposite, namely that users can combine the secret keys. To achieve collusion resistance, the scheme of [56] used a different polynomial Q to derive the secret key of each different user. To adapt it to an IBDT signature scheme, we define a single polynomial Q to derive the secret keys of all users that are associated to the same digit position of their identifiers; more specifically, in the adaptation of IBDT described in Section 3.3.5, a set of polynomials $\{Q_1, \ldots, Q_\ell\}$ is defined, where polynomial Q_j is used to derive the secret keys of all users that are associated to the j-th digit or group of digits of the users' identifiers.

Another change with respect to [56] is that our scheme is designed to work in groups with an asymmetric bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. For this reason, we need to embed the image of the hash function H in the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. This could be done in the standard model using two different copies of a Waters' hash function [95] in different groups, but the resulting scheme has very large parameters. Instead, we define a hash function which maps strings to elements of $\mathbb{Z}_p$ and we then embed the image of H in each of the groups with an affine function, inspired by [21]. For the security analysis, we treat H as a random oracle.

Additionally, we note that, for the group discount application, we only need s-out-of-s threshold policies in our IBDT signature scheme. This results in a slightly simpler scheme. Further, we define the space of identities $\mathcal{ID}$ as the set of all integers in the interval $[1, \ldots, p/2]$, where p is the order of the group in which the signature is defined.

▶ Setup $(1^\lambda, \mathcal{ID}, n)$: The algorithm chooses bilinear groups $\mathcal{G} = (g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order $p > 2^\lambda$, where g_i is a generator of $\mathbb{G}_i$ and a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$. The resulting public parameters are $\mathsf{pms} = (\mathcal{ID},\ n,\ \lambda,\ \mathcal{G},\ H)$, where the space of identities $\mathcal{ID}$ is the set of all integers in $[1, p/2]$. Then the algorithm randomly chooses $\alpha, \alpha_0 \leftarrow \mathbb{Z}_p$, $\vec{\alpha} = (\alpha_1, \ldots, \alpha_N)^\top \leftarrow \mathbb{Z}_p^N$, where $N = n + 1$. It then computes $e(g_1, g_2)^\alpha$, $h_0 = g_1^{\alpha_0}$, $f_0 = g_2^{\alpha_0}$, $\vec{H} = (h_1, \ldots, h_N)^\top = g_1^{\vec{\alpha}}$ and $\vec{F} = (f_1, \ldots, f_N)^\top = g_2^{\vec{\alpha}}$. Further, it defines a polynomial $Q[X] := \alpha + \beta_1 X + \cdots + \beta_{n-1} X^{n-1}$ where $\beta_1, \ldots, \beta_{n-1} \leftarrow \mathbb{Z}_p$. It also chooses $n-1$ arbitrary integers $\mathcal{D} = \{d_1, \ldots, d_{n-1}\} \in [(p+1)/2, p-1]$, for example, $d_i := (p+1)/2 + (i-1)$. For any $1 \le i \le n-1$, $\mathcal{D}_i$ denotes the first i elements in $\mathcal{D}$.

Finally, the algorithm picks $\tilde{u}_0, \tilde{u}_1 \leftarrow \mathbb{Z}_p$ and defines $\vec{U} := (g_1^{\tilde{u}_0}, g_1^{\tilde{u}_1})$ and $\vec{V} := (g_2^{\tilde{u}_0}, g_2^{\tilde{u}_1})$.

The master secret key is defined to be $\mathsf{msk} = (g_1^{\alpha}, Q)$ and the master public key is

$$\mathsf{mpk} = \Big(e(g_1, g_2)^{\alpha},\ h_0,\ f_0,\ \vec{H}, \vec{F}, \mathcal{D}, \vec{U}, \vec{V}\Big).$$

▶ $\mathsf{Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \mathbf{id})$: This algorithm generates a key component $\mathsf{SK}_{\mathbf{id}} = (D_{\mathbf{id},1}, D_{\mathbf{id},2}, K_{\mathbf{id},1}, \ldots, K_{\mathbf{id},N-1}, \{D_{\mathbf{id},j,1}, D_{\mathbf{id},j,2}, K_{\mathbf{id},j,1}, \ldots, K_{\mathbf{id},j,N-1}\}_{j=1\ldots n-1})$ by picking fresh random elements $r_{\mathbf{id}}, r_{\mathbf{id},1}, \ldots, r_{\mathbf{id},n-1} \leftarrow \mathbb{Z}_p$ and setting

$$\begin{aligned}
&D_{\mathbf{id},1} = g_1^{Q(\mathbf{id})} \cdot h_0^{r_{\mathbf{id}}},\\
&D_{\mathbf{id},2} = g_1^{r_{\mathbf{id}}},\\
&\Big\{K_{\mathbf{id},i} = \big(h_1^{-\mathbf{id}^i} \cdot h_{i+1}\big)^{r_{\mathbf{id}}}\Big\}_{i=1,\ldots,N-1},\\
&\Big\{D_{\mathbf{id},j,1} = g_1^{Q(d_j)} \cdot h_0^{r_{\mathbf{id},j}}, D_{\mathbf{id},j,2} = g_1^{r_{\mathbf{id},j}},\\
&\quad\big\{K_{\mathbf{id},j,i} = \big(h_1^{-d_j^i} \cdot h_{i+1}\big)^{r_{\mathbf{id},j}}\big\}_{i=1,\ldots,N-1}\Big\}_{j=1,\ldots,n-1}.
\end{aligned} \tag{3.1}$$

▶ $\mathsf{Sign}(\mathsf{pms}, \mathsf{mpk}, SK_{\mathbf{id}}, \mathsf{Msg}, \Gamma, \mathbf{id})$: To partially sign $\mathsf{Msg} \in \{0,1\}^*$ w.r.t. the policy $\Gamma = (s, S)$, where S is a set of identities of size $s = |S| \leq n$, the algorithm first computes $M = H(\mathsf{Msg}, \Gamma)$. It defines $\vec{Y} = (y_1, \ldots, y_N)^{\top}$ as the vector containing the coefficients of the polynomial

$$P_S(Z) = \sum_{i=1}^{N} y_i Z^{i-1} = \prod_{\mathbf{id} \in S} (Z - \mathbf{id}) \prod_{d \in \mathcal{D}_{n-s}} (Z - d). \tag{3.2}$$

Since $P_S(Z)$ is of degree n, it has at most $N = n+1$ non-zero coefficients. Then the algorithm sets

$$D'_{\mathbf{id},1} = D_{\mathbf{id},1} \cdot \prod_{i=1}^{N-1} K_{\mathbf{id},i}^{y_{i+1}} = g_1^{Q(\mathbf{id})} \cdot \big(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\big)^{r_{\mathbf{id}}}. \tag{3.3}$$

Let $M = H(\mathsf{Msg}, \Gamma) \in \mathbb{Z}_p$. Choose $z_{\mathbf{id}}, w_{\mathbf{id}} \leftarrow \mathbb{Z}_p$ and compute

$$\begin{gathered}
\sigma_{\mathbf{id},1} = D'_{\mathbf{id},1} \cdot \big(h_0 \cdot \textstyle\prod_{i=1}^{N} h_i^{y_i}\big)^{w_{\mathbf{id}}} \cdot \big(u_0^M \cdot u_1\big)^{z_{\mathbf{id}}},\\
\sigma_{\mathbf{id},2} = D_{\mathbf{id},2} \cdot g_1^{w_{\mathbf{id}}}, \sigma_{\mathbf{id},3} = g_1^{z_{\mathbf{id}}}.
\end{gathered}$$

Return the signature $\sigma_{\mathbf{id}} = (\sigma_{\mathbf{id},1}, \sigma_{\mathbf{id},2}, \sigma_{\mathbf{id},3}) \in \mathbb{G}_1^3$.

▶ $\mathsf{Comb}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \{\sigma_{\mathbf{id}}\}_{\mathbf{id}\in S}, \mathsf{SK}_{\mathbf{id}'})$: Given his secret key $\mathsf{SK}_{\mathbf{id}'} = (D_{\mathbf{id}',1}, D_{\mathbf{id}',2}, K_{\mathbf{id}',1}, \ldots, K_{\mathbf{id}',N-1}, \{D_{\mathbf{id}',j,1}, D_{\mathbf{id}',j,2}, K_{\mathbf{id}',j,1}, \ldots, K_{\mathbf{id}',j,N-1}\}_{j=1,\ldots,n-1})$, the combiner does the following operations:

1. For each $j = 1, \ldots, n-s$, compute:

$$\sigma'_{\mathbf{id}',j} := D_{\mathbf{id}',j,1} \cdot \prod_{i=1}^{N-1} K_{\mathbf{id}',j,i}^{y_{i+1}}.$$

2. Then compute:

$$\begin{aligned}
\sigma_1 &= \prod_{\mathbf{id}\in S} \left(\sigma_{\mathbf{id},1}\right)^{\Delta_{\mathbf{id}}^{S\cup\mathcal{D}_{n-s}}(0)} \prod_{j=1}^{n-s} \left(\sigma'_{\mathbf{id}',j}\right)^{\Delta_{d_j}^{S\cup\mathcal{D}_{n-s}}(0)} \\
&= g_1^{\alpha} \cdot (h_0 \cdot \prod_{i=1}^{N} h_i^{y_i})^r \cdot (u_0^M \cdot u_1)^z, \\
\sigma_2 &= \prod_{\mathbf{id}\in S} \left(\sigma_{\mathbf{id},2}\right)^{\Delta_{\mathbf{id}}^{S\cup\mathcal{D}_{n-s}}(0)} \prod_{j=1}^{n-s} \left(D_{\mathbf{id}',j,2}\right)^{\Delta_{d_j}^{S\cup\mathcal{D}_{n-s}}(0)} = g_1^r, \\
\sigma_3 &= \prod_{\mathbf{id}\in S_t} \left(\sigma_{\mathbf{id},3}\right)^{\Delta_{\mathbf{id}}^{S\cup\mathcal{D}_{n-s}}(0)} = g_1^z,
\end{aligned}$$

where $r = \sum_{\mathbf{id}\in S} \Delta_{\mathbf{id}}^{S\cup\mathcal{D}_{n-s}}(0) \cdot (r_{\mathbf{id}} + w_{\mathbf{id}}) + \sum_{j=1}^{n-s} \left(\Delta_{d_j}^{S\cup\mathcal{D}_{n-s}}(0) \cdot r_{\mathbf{id}',j}\right)$ and $z = \sum_{\mathbf{id}\in S} \Delta_{\mathbf{id}}^{S\cup\mathcal{D}_{n-s}}(0) \cdot z_{\mathbf{id}}$.
Return the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in \mathbb{G}_1^3$.

▶ $\mathsf{Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \sigma, \Gamma)$: This algorithm parses Γ as a pair (s, S). It computes $M = H(\mathsf{Msg}, \Gamma)$. Then, it defines the vector $\vec{Y} = (y_1, \ldots, y_N)^\top$ from the polynomial $P_S(Z)$ as per Equation (3.2). The algorithm accepts the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ as valid and thus outputs 1 if and only if

$$e(g_1, g_2)^{\alpha} \stackrel{?}{=} \frac{e(\sigma_1, g_2)}{e\big(\sigma_2, f_0 \cdot \prod_{i=1}^{N} f_i^{y_i}\big) \cdot e\big(\sigma_3, (v_0^M \cdot v_1)\big)}. \tag{3.4}$$

3.3.4 Security of the IBDT instance

Theorem 1. *The IBDT signature scheme of Section 3.3.3 is selective-predicate and adaptive-message unforgeable under chosen-message attacks if H is collision-resistant and if the asymmetric $(n+1)$-DHE assumption holds in the bilinear*

group $\mathcal{G}$, where n is an upper bound of the number of signers s in any threshold policy.

Proof

We show that a forger $\mathcal{F}$ implies either a collision-finder for H or an algorithm $\mathcal{B}$ that computes $g^{\gamma^{N+1}}$ from $(g_1, g_2, g_1^{\gamma}, \ldots, g_1^{\gamma^N}, g_1^{\gamma^{N+2}}, \ldots, g_1^{\gamma^{2N}}, g_2^{\gamma}, \ldots, g_2^{\gamma^N})$, where $N = n+1$. In the following, we denote by $\vec{\gamma}$ the vector $\vec{\gamma} := (\gamma, \gamma^2, \ldots, \gamma^N)$ and by z_i the value $z_i := g_1^{\gamma^i}$, for each $i \in \{1, \ldots, 2N\}$. Also, we assimilate a user's identity-based public key with her identity and denote it as **id**.

At the outset of the attack game, $\mathcal{F}$ declares the challenge set $\Gamma^\star = (s^\star, S^\star)$. Then $\mathcal{B}$ prepares the public parameters pms and the master public key mpk as follows: it selects a set $\mathcal{D}$ of n dummy signers and computes the vector $\vec{Y}$ associated with the polynomial $P_{S^\star}(Z)$ according to Expression (3.2) using the set $\mathcal{D}_{n-s^\star}$ of the first $n-s^\star$ dummy signers. More precisely, $\mathcal{B}$ picks $\theta_0, \delta_0 \leftarrow \mathbb{Z}_p$ and a random vector $\vec{\theta} \leftarrow \mathbb{Z}_p^N$ and computes $\vec{H} = (h_1, \ldots, h_N)^\top = g_1^{\vec{\gamma}} \cdot g_1^{\vec{\theta}}$, $\vec{F} = (f_1, \ldots, f_N)^\top = g_2^{\vec{\gamma}} \cdot g_2^{\vec{\theta}}$ (which implicitly sets $\vec{\alpha} = \vec{\gamma} + \vec{\theta}$), $h_0 = g_1^{\theta_0} \cdot g_1^{-\langle \vec{\gamma}, \vec{Y} \rangle}$, $f_0 = g_2^{\theta_0} \cdot g_2^{-\langle \vec{\gamma}, \vec{Y} \rangle}$ and $e(g_1, g_2)^\alpha = e(z_N, g_2^{\gamma})^{\delta_0}$; the master secret key (implicitly) is set to $g_1^\alpha = z_{N+1}^{\delta_0}$. In addition, $\mathcal{B}$ selects a collision-resistant hash function H which will be treated as a random oracle. It chooses some value $M^\dagger \leftarrow \mathbb{Z}_p$ and stores it to answer random oracle queries. It also defines $u_0 := z_1^{t_0}$, $v_0 := (g_2^{\gamma})^{t_0}$, $u_1 := z_1^{-M^\dagger t_0} g_1^{t_1}$, $v_1 := (g_2^{\gamma})^{-M^\dagger t_0} g_2^{t_1}$ for $t_0, t_1 \leftarrow \mathbb{Z}_p$. The master public key $\mathsf{mpk} = \big(e(g_1, g_2)^\alpha, h_0, f_0, \vec{H}, \vec{F}, \vec{U}, \vec{V}, \mathcal{D}, H\big)$ is given to $\mathcal{F}$.

In the following, for any $\omega \in \mathbb{Z}_p$, we define the vector $\vec{X}_\omega^n = (1, \omega, \ldots, \omega^{n-1})^\top$. We note that, given any set $S \subset \mathbb{Z}_p$ of cardinality less than n, the vectors $\{\vec{X}_\omega^n\}_{\omega \in S}$ are linearly independent.

Random oracle queries: We assume that the number of random oracle queries of an adversary is bounded by some natural number q_H. Algorithm $\mathcal{B}$ chooses a random index $i^\star \in [q_H]$ and answers the $i^\star$-th query with the value $M^\dagger \in \mathbb{Z}_p$, and the other queries with randomly chosen elements in $\mathbb{Z}_p$.

Secret key queries: $\mathcal{F}$ can obtain secret keys for any identity-based public key, provided that the set of queried identities Ω is such that $|\Omega \cap S^\star| < s^\star$. Since $|\Omega \cap S^\star| < s^\star$, and $S^\star$ and $\mathcal{D}$ are disjoint sets of identity-based public keys (just like Ω and $\mathcal{D}$), the cardinality of $(S^\star \cap \Omega) \cup \mathcal{D}_{n-s^\star}$ is strictly less than n. Consequently, the vector $\vec{X}_0^n = (1, 0, \ldots, 0)^\top$ cannot be in the span of the

vectors $\{\vec{X}_\omega^n\}_{\omega\in(S^\star\cap\Omega)\cup\mathcal{D}_{n-s^\star}}$. Pick $\mu \leftarrow \mathbb{Z}_p^*$. We conclude that there exists an efficiently computable vector $\vec{\tau}$ which is uniform conditioned on $\langle\vec{X}_\omega^n, \vec{\tau}\rangle = 0$ for any $\omega \in (S^\star \cap \Omega) \cup \mathcal{D}_{n-s^\star}$ and $\langle\vec{X}_0^n, \vec{\tau}\rangle = \mu \neq 0$ (according to Proposition 1 in [54]).
To construct a secret key, $\mathcal{B}$ has to define a random vector $\vec{u}$ which satisfies the constraint $\langle\vec{X}_0^n, \vec{u}\rangle = \alpha$, *i.e.* $\vec{u} = (\alpha, \beta_1, \ldots, \beta_{n-1})^\top$. This vector defines the coefficients of $Q[X]$. To this end, $\mathcal{B}$ proceeds as in the proof of Theorem 3 in [54], by implicitly setting $\vec{u}$ as $\vec{u} = \vec{v} + \psi \cdot \vec{\tau}$, where $\vec{v} = (v_1, \ldots, v_n)^\top \in \mathbb{Z}_p^n$ is a randomly chosen vector and $\psi = (\alpha - v_1)/\mu$, so that $\langle\vec{X}_0^n, \vec{u}\rangle = \alpha$. The task of $\mathcal{B}$ is thus to compute (without knowing the vector $\vec{u}$) a secret key component

$$\big(D_{\mathbf{id},1}, D_{\mathbf{id},2}, \{K_{\mathbf{id},i}\}_{i=1}^{N-1}\big) = \big(g_1^{Q(\mathbf{id})} \cdot h_0^{r_{\mathbf{id}}}, g_1^{r_{\mathbf{id}}}, \{h_1^{-\mathbf{id}^i} h_{i+1}\}_{i=1}^{N-1}\big),$$

and

$$\begin{aligned}&\{D_{\mathbf{id},j,1}, D_{\mathbf{id},j,2}, \{K_{\mathbf{id},j,i}\}_{i=1}^{N-1}\}_{j=1\ldots n-1} = \\ &= \{g_1^{Q(d_j)} \cdot h_0^{r_{\mathbf{id},j}}, g_1^{r_{\mathbf{id},j}}, \{h_1^{-d_j^i} h_{i+1}\}_{i=1}^{N-1}\}_{j=1\ldots n-1},\end{aligned}$$

where $Q(\omega) = \langle\vec{X}_\omega^n, \vec{u}\rangle$, for any $\omega \in \Omega \cup \mathcal{D}$.
We first explain how to compute the first row of each secret key, *i.e.* $\big(D_{\mathbf{id},1}$, $D_{\mathbf{id},2}$, $\{K_{\mathbf{id},i}\}_{i=1}^{N-1}\big)$.

1. For each $\mathbf{id} \in S^*$, we have $Q(\mathbf{id}) = \langle\vec{X}_{\mathbf{id}}^n, \vec{u}\rangle = \langle\vec{X}_{\mathbf{id}}^n, \vec{v}\rangle$ which is efficiently computable by $\mathcal{B}$. Hence, $\mathcal{B}$ can simply pick $r_{\mathbf{id}} \leftarrow \mathbb{Z}_p^*$ and define

$$D_{\mathbf{id}} = \big(D_{\mathbf{id},1}, D_{\mathbf{id},2}, \{K_{\mathbf{id},i}\}_{i=1}^{N-1}\big) = \Big(g_1^{Q(\mathbf{id})} \cdot h_0^{r_{\mathbf{id}}},\ g_1^{r_{\mathbf{id}}},\ \{(h_1^{-\mathbf{id}^i} h_{i+1})^{r_{\mathbf{id}}}\}_{i=1}^{N-1}\Big).$$

2. For each $\mathbf{id} \in \Omega\backslash\{S^*\}$, $\mathcal{B}$ can construct a valid key tuple $\big(D_{\mathbf{id},-1}, D_{\mathbf{id},2}$, $\{K_{\mathbf{id},i}\}_{i=1}^{N-1}\big)$ in two steps. The first step consists in building a tuple of the form

$$\big(D_{\mathbf{id},1}^*, D_{\mathbf{id},2}^*, \{K_{\mathbf{id},i}^*\}_{i=1}^{N-1}\big) = \Big(g_1^\alpha \cdot h_0^{\tilde{r}_{\mathbf{id}}},\ g_1^{\tilde{r}_{\mathbf{id}}},\ \{(h_1^{-\mathbf{id}^i} h_{i+1})^{\tilde{r}_{\mathbf{id}}}\}_{i=1}^{N-1}\Big)$$

using the fact that $\mathbf{id}$ is not in $S^\star \cup \mathcal{D}_{n-s^\star}$. To this end, $\mathcal{B}$ proceeds as in [25]. Let $M_{\mathbf{id}} \in \mathbb{Z}_p^{N\times(N-1)}$ be the matrix $M_{\mathbf{id}} = \begin{pmatrix} -\mathbf{id} & -\mathbf{id}^2 & \cdots & -\mathbf{id}^{N-1} \\ & I_{N-1} & & \end{pmatrix}$. Pick $\xi_1 \leftarrow \mathbb{Z}_p^*$ and define $\vec{\xi} = \xi_1 \cdot (1, \mathbf{id}, \ldots, \mathbf{id}^{N-1})^\top$, which satisfies

$\vec{\xi}^{\top} M_{\mathbf{id}} = \vec{0}$ while $\langle \vec{Y}, \vec{\xi} \rangle = \xi_1 \cdot P_{S^\star}(\mathbf{id}) \neq 0$. The simulator $\mathcal{B}$ computes

$$\begin{aligned} \big(D^*_{\mathbf{id},1}, D^*_{\mathbf{id},2}\big) &= \Big(g_1^{\alpha} \cdot h_0^{\tilde{r}_{\mathbf{id}}},\ g_1^{\tilde{r}_{\mathbf{id}}}\Big) \\ \text{and} \qquad \big(K^*_{\mathbf{id},1}, \ldots, K^*_{\mathbf{id},N-1}\big)^{\top} &= g_1^{\tilde{r}_{\mathbf{id}} M_{\mathbf{id}}^{\top} \vec{\alpha}}, \end{aligned} \tag{3.5}$$

with $\vec{\alpha} = (\alpha_1, \ldots, \alpha_N)^{\top}$ and where the exponent $\tilde{r}_{\mathbf{id}}$ is defined as $\tilde{r}_{\mathbf{id}} = r + \delta_0 \langle (\gamma^N, \gamma^{N-1}, \ldots, \gamma)^{\top}, \vec{\xi} \rangle / \langle \vec{Y}, \vec{\xi} \rangle$ for some $r \leftarrow \mathbb{Z}_p$ chosen by $\mathcal{B}$. Since $g^{M_{\mathbf{id}}^{\top} \vec{\alpha}} = (h_1^{-\mathbf{id}} h_2, \ldots, h_1^{-\mathbf{id}^{N-1}} h_N)^{\top}$, if we can argue that both expressions in (3.5) are computable by $\mathcal{B}$, we will have concluded the first step.
For any $\vec{x} \in \mathbb{Z}_p^N$, the coefficient of γ^{N+1} in the product $\tilde{r}_{\mathbf{id}} \langle \vec{x}, \vec{\gamma} \rangle$ is $\delta_0 \langle \vec{x}, \vec{\xi} \rangle / \langle \vec{Y}, \vec{\xi} \rangle$. The reason why $\mathcal{B}$ can compute the second factor of $D^*_{\mathbf{id},1}$ in (3.5) is that the coefficient of $g^{\gamma^{N+1}}$ in $D^*_{\mathbf{id},1}$ is 0. Indeed, $D^*_{\mathbf{id},1} = g^{\alpha} \cdot h_0^{\tilde{r}_{\mathbf{id}}} = z_{N+1}^{\delta_0} \cdot \big(g^{\theta_0} \cdot g^{-\langle \vec{\gamma}, \vec{Y} \rangle}\big)^{\tilde{r}_{\mathbf{id}}}$ and the coefficient of γ^{N+1} is $-\delta_0$ in the product $-\tilde{r}_{\mathbf{id}} \langle \vec{\gamma}, \vec{Y} \rangle$, as we can see by applying the observation above in the case $\vec{x} = \vec{Y}$. Since $M_{\mathbf{id}}^{\top} \vec{\xi} = \vec{0}$, by applying the above observation to the case where $\vec{f}^{\top}$ is successively set as the rows of $M_{\mathbf{id}}^{\top}$, we find that $z_{N+1} = g_1^{\gamma^{N+1}}$ does not appear in $g_1^{\tilde{r}_j \cdot M_{\mathbf{id}}^{\top} \vec{\alpha}}$, which is computable.
This concludes the first step of the key generation process. In the second step, we just have to turn $\big(D^*_{\mathbf{id},1}, D^*_{\mathbf{id},2}, \{K^*_{\mathbf{id},i}\}_{i=1}^{N-1}\big)$ into a suitable key component. Note that

$$\begin{aligned} \langle \vec{X}^n_{\mathbf{id}}, \vec{u} \rangle &= \langle \vec{X}^n_{\mathbf{id}}, \vec{v} \rangle + \psi \cdot \langle \vec{X}^n_{\mathbf{id}}, \vec{\tau} \rangle \\ &= \sum_{j=1}^{n} \mathbf{id}^{j-1} \Big(v_j + \frac{(\alpha - v_1)}{\mu} \cdot \tau_j \Big) = \kappa_1 \cdot \alpha + \kappa_2, \end{aligned}$$

where $\kappa_1 = (\sum_{j=1}^{n} \mathbf{id}^{j-1} \tau_j) \cdot \mu^{-1}$ and $\kappa_2 = \mu^{-1} \cdot \sum_{j=1}^{n} \mathbf{id}^{j-1} \big(\mu v_j - v_1 \tau_j\big)$ are computable, so that $\mathcal{B}$ can obtain a well-formed tuple $(D_{\mathbf{id},1}, D_{\mathbf{id},2}, \{K_{\mathbf{id},i}\}_{i=1}^{N-1})$ by picking $r'_j \leftarrow \mathbb{Z}_p$ and setting

$$\begin{aligned} \mathsf{SK}_{\mathbf{id}} =& (D_{\mathbf{id},1}, D_{\mathbf{id},2}, \{K_{\mathbf{id},i}\}_{i=1}^{N-1}) \\ =& \Big(D^*_{\mathbf{id},1}{}^{\kappa_1} \cdot g_1^{\kappa_2} \cdot h_0^{r'_{\mathbf{id}}}, D^*_{\mathbf{id},2}{}^{\kappa_1} \cdot g_1^{r'_{\mathbf{id}}}, \{K^*_{\mathbf{id},i}{}^{\kappa_1} \cdot (h_1^{-\mathbf{id}^i} \cdot h_{i+1})^{r'_{\mathbf{id}}}\}_{i=1}^{N-1} \Big). \end{aligned}$$

Finally, we just have to argue how to compute, for each $\mathbf{id} \in \Omega$ and for each d_j, $j = 1, \ldots, n-1$, the rest of the components of the secret key, namely: $\{D_{\mathbf{id},j,1}, D_{\mathbf{id},j,2}, \{K_{\mathbf{id},j,i}\}_{i=1}^{N-1}\}_{j=1,\ldots,n-1}$. The analysis is the same as before,

namely, for each $j = 1, \ldots, n - s^*$, the tuple is computed as in the first item (*i.e.* as in the case where $\mathbf{id} \in S^*$) and for each $j = n - s^* + 1, \ldots, n - 1$ as in the second item (using independent randomness for each id).

Signing queries: At any time, $\mathcal{F}$ is also allowed to obtain signatures on arbitrary messages. At each signing query, $\mathcal{F}$ supplies a message Msg and a threshold access policy $\Gamma = (s, S)$, where S is a set of identities of size $s \leq n$. To answer such a query, $\mathcal{B}$ computes $M = H(\mathsf{Msg}, \Gamma) \in \mathbb{Z}_p$ by checking the list of its random oracle queries and aborts if $M = M^\dagger$. Else, $\mathcal{B}$ constructs the vector $\vec{Y} = (y_1, \ldots, y_N)^\top$ whose coordinates are the coefficients of the polynomial $P_S(Z)$ which is obtained following Expression (3.2), by augmenting S with $n - s$ dummy signers. Recall that $\mathcal{B}$ has to generate a signature of the form

$$\sigma_1 = g_1^\alpha \cdot \big(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\big)^r \cdot (u_0^M \cdot u_1)^{\tilde{z}}, \quad \sigma_2 = g_1^r, \quad \sigma_3 = g_1^{\tilde{z}}, \tag{3.6}$$

for some $r, \tilde{z} \leftarrow \mathbb{Z}_p$. To this end, $\mathcal{B}$ uses the usual technique (which dates back to [21]) consisting in implicitly defining $\tilde{z} = z + \dfrac{\gamma^N \cdot \delta_0}{t_0 \cdot (M - M^\dagger)}$, for a randomly chosen $z \leftarrow \mathbb{Z}_p$, and computing

$$\sigma_1 = \big(u_0^M \cdot u_1\big)^z \cdot z_N^{\frac{t_1 \delta_0}{t_0 (M - M^\dagger)}} \cdot \big(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\big)^r,$$

$$\sigma_2 = g_1^r, \quad \sigma_3 = g_1^z \cdot z_N^{\frac{\delta_0}{t_0 \cdot (M - M^\dagger)}},$$

for a random $r \leftarrow \mathbb{Z}_p$. Since α is implicitly defined as $\alpha = \gamma^{N+1} \cdot \delta_0$, the above triple is easily seen to have the required distribution (3.6).

Forgery: The adversary eventually outputs a forgery $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star)$ for some message $\mathsf{Msg}^\star$ and the target access policy $\Gamma^\star = (s^\star, S^\star)$. At this step, $\mathcal{B}$ computes $M^\dagger = H(\mathsf{Msg}^\star, \Gamma^\star)$ by checking its list of oracle queries. It aborts if it holds that either:

1. The hash value $M^\star = H(\mathsf{Msg}^\star, \Gamma^\star)$ is not equal to $M^\dagger$;

2. $\mathcal{F}$ made a signing query (Msg, Γ) such that $(\mathsf{Msg}, \Gamma) \neq (\mathsf{Msg}^\star, \Gamma^\star)$ and $H(\mathsf{Msg}, \Gamma) = H(\mathsf{Msg}^\star, \Gamma^\star)$.

Case 2 cannot occur under the assumption that H is collision-resistant. Conditioned on Case 2 not occurring, the complementary event of Case 1 occurs with probability $1/q_H$.

Assuming the adversary $\mathcal{B}$ does not abort, it can compute $z_{N+1} = g_1^{\gamma^{N+1}}$ as follows. Since the vector $\vec{Y} = (y_1, \ldots, y_N)^\top$ derived from $\Gamma^\star$ is such that $f_0 \cdot \prod_{i=1}^{N} f_i^{y_i} = g_2^{\theta_0 + \langle \vec{\theta}, \vec{Y} \rangle}$, $v_0^M \cdot v_1 = v_0^{M^\dagger} \cdot v_1 = g_2^{t_1}$, and $\sigma^\star$ satisfies Equation (3.4), we must have

$$e(g_1, g_2)^{(\gamma^{N+1}) \cdot \delta_0} = \frac{e(\sigma_1^\star, g_2)}{e(\sigma_2^\star, g_2^{\theta_0 + \langle \vec{\theta}, \vec{Y} \rangle}) \cdot e(\sigma_3^\star, g_2^{t_1})}.$$

This implies that $z_{N+1} = \left(\frac{\sigma_1^\star}{(\sigma_2^\star)^{\theta_0 + \langle \vec{\theta}, \vec{Y} \rangle} \cdot (\sigma_3^\star)^{t_1}} \right)^{1/\delta_0}$, which is computable by $\mathcal{B}$. □

3.3.5 Choice of parameters for implementation

We discuss here how to choose parameters for the above IBDT signature scheme in the context of its application to group discounts and, more specifically, high-occupancy vehicle toll discounts:

- Choice of λ: this is just the security parameter, which should specify the size of the bilinear group (see recommendations in [69]).
- Choice of n: this corresponds to the maximum number of group members. In the case of HOVs, it would be the maximum number of passengers that can travel in a car (that is, typically n is around 9 or 10).
- Each identity-based public key used in the IBDT signature scheme is derived from a group of η digits in the national identity card (or another identifier) of a user **id**. As discussed in Section 3.4 below, η is a parameter that trades off accomodating larger values of n against anonymity (the larger η, the larger n can be, but the less anonymity).

As we announced, we will make a particular use of an IBDT signature scheme, linked to a special key management which will provide anonymity and which is explained in the next section. We begin by splitting the identity space $\mathcal{ID}$ into ℓ disjoint subspaces whose respective elements are clearly recognizable as belonging to a certain subspace only, *i.e.* $\mathcal{ID} = \mathcal{ID}_1 \cup \ldots \cup \mathcal{ID}_\ell$, where ℓ is a functionality and anonymity parameter (related to the length of the users'

identifiers, see Section 3.4), and $\mathcal{ID}_j$ is the subspace associated to the j-th group of η digits in the users' identifiers, for $j = 1, \ldots, \ell$. In our use of an IBDT scheme for the group discount application, every user **id** receives the secret keys associated with a vector of identity-based public keys $\{\mathsf{ik}_1^{\mathbf{id}}, \ldots, \mathsf{ik}_\ell^{\mathbf{id}}\}$, with $\mathsf{ik}_j^{\mathbf{id}} \in \mathcal{ID}_j$, unlike the straightforward approach of previous sections in which a user was assigned a single identity-based public key denoted just **id** like the user's identifier. Thus, in our protocol, to sign a message on behalf of s users $\mathbf{id}_1, \ldots, \mathbf{id}_s$, they must all agree on an index j in $\{1, \ldots, \ell\}$ and sign using $\mathsf{ik}_j^{\mathbf{id}_1}, \ldots, \mathsf{ik}_j^{\mathbf{id}_s}$ (the whole discussion on how to do this can be found in Section 3.4 below). As a consequence, the IBDT has to be tuned to fit this special key management structure. The modification is straightforward and is fully discussed below.

Given a user **id** who requests the secret keys for a set of identity-based public keys $\mathbf{IK_{id}} = \{\mathsf{ik}_1^{\mathbf{id}}, \ldots, \mathsf{ik}_\ell^{\mathbf{id}}\}$, the secret keys that he receives are essentially just ℓ secret keys of the original IBDT scheme, except that the keys are randomized to make sure that the secret key for identity-based public key $\mathsf{ik}_j^{\mathbf{id}}$, for any $j = 1, \ldots, \ell$ can only be used with secret keys for the j-th identity-based public keys of other users. A straightforward application of this idea would mean that each user receives a secret key which is ℓ times the size of a single IBDT key. To increase efficiency, some parts of the keys are reused. More specifically, below we give a detailed account on how to adapt the IBDT scheme to our key management design when $\ell > 1$.

Note that we implicitly assume that, for all $j \neq j'$, the values of the j-th and the j'-th component of vector $\mathbf{IK_{id}}$ are taken from two disjoint and easily recognizable sets. This is the case for instance if the j-th component of $\mathbf{IK_{id}}$ is of the form $j||d_j$, as we suggested.

▶ Setup $(1^\lambda, \mathcal{ID}, n, \ell)$: The only change here is that now identities are ℓ-dimensional vectors of elements in $[p-1/2]$, denoted by $\mathbf{ID_{id}}$, and that the master secret key is $\mathsf{msk} = (g_1^\alpha, Q_1, \ldots, Q_\ell)$, where $Q_i[X]$ are polynomials in $\mathbb{Z}_p[X]$ of degree $n-1$ chosen uniformly independently and uniformly at random subject to $Q_i(0) = \alpha$. Note that mpk is unchanged.

▶ $\mathsf{Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \mathbf{IK_{id}})$: This algorithm generates a key component $\mathsf{SK_{id}} := (\{D_{\mathbf{id},1,k}\}_{k=1}^{\ell}, D_{\mathbf{id},2}, K_{\mathbf{id},1}, \ldots, K_{\mathbf{id},N-1}, \{\{D_{\mathbf{id},j,1,k}\}_{k=1}^{\ell}, D_{\mathbf{id},j,2}, K_{\mathbf{id},j,1}, \ldots, K_{\mathbf{id},j,N-1}\}_{j=1\ldots n-1})$ by picking fresh random elements

$r_{\mathbf{id}}, r_{\mathbf{id},1}, \ldots, r_{\mathbf{id},n-1} \leftarrow \mathbb{Z}_p$ and setting:

$$\begin{aligned}
&\{D_{\mathbf{id},1,k} = g_1^{Q_k(\mathbf{id})} \cdot h_0^{r_{\mathbf{id}}}\}_{k=1}^{\ell},\\
&D_{\mathbf{id},2} = g_1^{r_{\mathbf{id}}},\\
&\left\{K_{\mathbf{id},i} = \left(h_1^{-\mathbf{id}^i} \cdot h_{i+1}\right)^{r_{\mathbf{id}}}\right\}_{i=1,\ldots,N-1},\\
&\left\{\{D_{\mathbf{id},j,1,k} = g_1^{Q_k(d_j)} \cdot h_0^{r_{\mathbf{id},j}}\}_{k=1}^{\ell}, D_{\mathbf{id},j,2} = g_1^{r_{\mathbf{id},j}},\right.\\
&\left.\left\{K_{\mathbf{id},j,i} = \left(h_1^{-d_j^i} \cdot h_{i+1}\right)^{r_{\mathbf{id},j}}\right\}_{i=1,\ldots,N-1}\right\}_{j=1,\ldots,n-1}. \qquad (3.7)
\end{aligned}$$

Algorithm Sign is the same as specified in Section 3.3.3 except that messages are signed not for the whole vector of identity-based public keys $\mathbf{IK_{id}} = \{\mathsf{ik}_1^{\mathbf{id}}, \ldots, \mathsf{ik}_\ell^{\mathbf{id}}\}$ but for one single component $\mathsf{ik}_j^{\mathbf{id}}$. Similarly, algorithm Comb is the same as in the original IBDT, except that it takes as input a set of signatures for some identities $\mathsf{ik}_{j_1}^{\mathbf{id}_1}, \ldots, \mathsf{ik}_{j_s}^{\mathbf{id}_s}$ and combines them in a single signature in case $j_1 = \ldots = j_s$ and outputs $\perp$ otherwise.

Note that in terms of efficiency, with respect to the original IBDT scheme, users have to store $2(\ell-1)$ additional group elements as part as their secret key. The size of the public parameters and the cost of the signing and combining algorithms are unchanged.

The security proof of the original IBDT scheme can be trivially modified to prove the security of this scheme. Indeed, it suffices to define in the new proof $Q_1[X] := Q[X]$ and the rest of the polynomials $Q_j[X]$ as $Q_j := Q_1 + R_j$, for some polynomial R_j chosen uniformly at random subject to $R_j(0) = 0$. Since adversary $\mathcal{B}$ can choose the polynomials R_j on his own, it is obvious that $\mathcal{B}$ can simulate the secret keys for any vector of identity-based public keys $\mathbf{IK_{id}} = \{\mathsf{ik}_1^{\mathbf{id}}, \ldots, \mathsf{ik}_\ell^{\mathbf{id}}\}$ for all $\mathbf{IK_{id}}$ not in the challenge set S^* in the same way $\mathcal{B}$ simulated the secret keys for all identities not in S^* in the original IBDT proof.

3.4 Key management

Our accreditation mechanism provides anonymity to group members by employing specially crafted key generation and management protocols. Leaning on the properties of identity-based cryptography, that is, the possibility to use arbitrary strings as public keys, we develop a key generation and management protocol which renders users indistinguishable from many other users when signing with

an IBDT signature scheme. The amount of users from whom any single user is indistinguishable is determined by a system parameter η, described below.

In our protocol, every user $\mathbf{id}_i$ is given an ordered list of identity-based public keys $\mathbf{IK}_{\mathbf{id}_i}$ that depends on a unique identifier of the user, such as her national identity card number, her phone number, the IMEI (international mobile equipment identity) of her mobile device or a combination of any of them. This identifier is denoted as $\mathbf{id}_i = d_k^i d_{k-1}^i \dots d_1^i$ of length k, where d_j^i is the j-th last digit of $\mathbf{id}_i$ and typically ranges from 0 to 9.

The procedure to generate a list of identity-based public keys associated to an identifier $\mathbf{id}_i$ is as follows: first, choose a value $\ell < k$ and take the ℓ last digits of $\mathbf{id}_i$. Then for every digit $d_1^i, \dots, d_\ell^i$, build an identity-based public key $\mathsf{ik}_j^{d_j^i}$ as an encoding of the digit d_j^i and the position it occupies in $\mathbf{id}_i$, for example $\mathsf{ik}_j^{d_j^i} = j \,||\, d_j^i$, where $||$ is the concatenation operation. This results in a vector of identity-based public keys

$$\mathbf{IK}_{\mathbf{id}_i} = \left\{ \mathsf{ik}_1^{d_1^i}, \dots, \mathsf{ik}_\ell^{d_\ell^i} \right\}.$$

To illustrate this process, imagine $\mathbf{id}_i = 12345678$ and $\ell = 4$. The resulting public key list would be $\mathbf{IK}_{\mathbf{id}_i} = \{18, 27, 36, 45\}$.

Once the identities are generated, the certification authority (or any other trusted entity) generates and sends to each user $\mathbf{id}_i$ the secret keys corresponding to the set of identity-based public keys $\mathbf{IK}_{\mathbf{id}_i}$ generated with a modification of the IBDT scheme explained in Section 3.3.5.

To prove the number of members in a group, the members will choose a common integer $j \in \{1, \dots, \ell\}$ so that *the j-th digits in their identifiers (and hence their j-th identity-based public keys) are different for all of them.* Then they will use the IBDT signature scheme, using the corresponding secret keys, to certify the size of their group.

Assuming that the values of the digits range from 0 to 9, this would provide anonymity to each of the users, since on average 10% of people will share a certain value of the j-th digit for some value of j.

This approach limits the size of the groups that can be certified with our method to a maximum of 10, since it is impossible for more than 10 users to find an index j so that all digits in the j-th position are different. Moreover, intuition tells us that the closer the size of the group to this maximum size, the more difficult it becomes to find a value of j. Additionally, by the birthday paradox, the probability of finding colliding digits will be high even for group sizes far from the maximum size. The probability that our protocol fails depends

on the number of keys each user is given, ℓ, and the size of the group n; more specifically, for $n \leq 10$:

$$F(\ell, n) = \left(1 - \frac{10(10-1)\dots(10-n+1)}{10^n}\right)^{\ell},$$

that is very close to 1 for values of n close to 10.

This limitation can be partially solved by assigning $\eta \geq 2$ digits of $\mathbf{id}_i$ to each of the ℓ public keys, instead of just one digit. By doing this, the maximum value for the size of the groups becomes 10^η, and the probability of failure, for values of $n \leq 10^\eta$, is

$$F(\ell, n, \eta) = \left(1 - \frac{10^\eta(10^\eta-1)\dots(10^\eta-n+1)}{10^{\eta n}}\right)^{\ell}.$$

However, the price to be paid for choosing a larger η is a loss of anonymity, since, if more digits are associated to each identity-based public key, less users share that public key. For example, for $\eta = 2$ a user would share each of his identity-based public keys with only 1% of the total number of users.

The parameters ℓ and η, which impact on the number of keys each user stores and the anonymity level of users, are system parameters that the service provider can adjust as required.

We note that, with our scheme, two users $\mathbf{id}_1, \mathbf{id}_2$ can pool the sets of secret keys corresponding to their respective vectors of identity-based public keys $\mathbf{IK}_{\mathbf{id}_1}$ and $\mathbf{IK}_{\mathbf{id}_2}$ to create the secret keys corresponding to a vector of identity-based public keys $\mathbf{IK}_{\mathbf{id}_3}$. To do this, they only need to combine the secret keys corresponding to some of the identities of $\mathbf{id}_1$ and some of the identities of $\mathbf{id}_2$. However, this does not allow $\mathbf{id}_1$ and $\mathbf{id}_2$ to prove that at least three users have collaborated to create a signature, because there is no index j such that the j-th identity-based public key in $\mathbf{IK}_{\mathbf{id}_1}$, $\mathbf{IK}_{\mathbf{id}_2}$ and $\mathbf{IK}_{\mathbf{id}_3}$ takes more than two different values. More generally, if t users pool their sets of secret keys to fabricate a $t+1$-th vector of identity-based public keys, they cannot prove that they are at least $t+1$ users, because there is no index j such that the j-th identity-based public key in the $t+1$ vectors takes more than t different values. Hence, this key pooling attack is harmless for the purpose of our system, because it does not allow any group of users to prove that they are more people than they actually are.

3.5 Method to accredit the size of a group

The following elements are needed in order to implement our accreditation method:

- A smartphone application $\mathsf{App}_{\mathbf{id}}$ published by a service provider (SP), who, after some registration process, also distributes the public parameters and keys of an *IBDT* signature scheme Π to each user **id**. Specifically, the $\mathsf{App}_{\mathbf{id}}$ of user **id** must provide the following functionalities:
  - to compute signatures with Π on behalf of **id**;
  - to compute ciphertexts with a public-key encryption scheme Π' selected by SP, under SP's public key pk^{SP};
  - to be able to run in master or slave mode, which affects the role $\mathsf{App}_{\mathbf{id}}$ plays in the accreditation protocol;
  - to include some certificate that allows checking the validity of pk^{SP};
  - to be able to interact with the applications of the rest of the group members and the verifying devices using short-range communication technologies (specifically NFC with the rest of group members and Bluetooth with the verifying device).
- Prepaid scratch cards available at stores, to be used for payment. Each card includes a code $\mathsf{Pay.Code}$ that the SP associates with an account holding a fixed credit specified by the card denomination.
- Verifying devices located at suitable places in SP's infrastructures that can:
  - verify signatures with Π;
  - hold SP's certificates as well as the keys needed to decrypt ciphertexts produced with Π' under pk^{SP};
  - communicate within short range with the users' devices.
- A procedure to thwart or punish system misuse.

Next, we describe the accreditation protocol:

Protocol 1. ***System set-up protocol.***

1. *SP publishes the service terms and conditions, along with the registration procedure, which describes what user identifier is to be used as* $\mathbf{id}$ *and the values for* ℓ *and* η.

2. *SP computes the public parameters* pms, *the master public key* mpk *and the master secret key* msk *of an* IBDT *signature scheme* Π *as per Algorithm* $\mathsf{IBDT.Setup}$. *SP makes* pms *and* mpk *available.*

3. *SP generates the parameters of a public-key encryption scheme* Π' *and makes them publicly available.*

Protocol 2. ***Registration protocol.***

1. *A user with identifier* $\mathbf{id}$ *authenticates to the service provider, face-to-face or otherwise.* $\mathbf{id}$ *is given a PIN code* $\mathsf{pin}_{\mathbf{id}}$.

2. *SP associates a vector of public keys of* Π, *say* $\mathbf{IK_{id}}$, *with* $\mathbf{id}$, *in the way explained in Section 3.4.*

3. *SP computes the secret keys associated to* $\mathbf{IK_{id}}$ *as per Algorithm* $\mathsf{IBDT.Keygen}$:
$$\mathbf{SK_{id}} = \left(\mathsf{sk}_1^{d_1^{\mathbf{id}}}, \ldots, \mathsf{sk}_\ell^{d_\ell^{\mathbf{id}}}\right).$$

4. $\mathbf{id}$ *downloads the smartphone app* $\mathsf{App_{id}}$ *and, using* $\mathsf{pin_{id}}$, *completes the registration and obtains the system parameters and keys, as well as the public key* pk^{SP}.

Protocol 3. ***Credit purchase.***

1. *A user buys a prepaid scratch card for the system at a store.*

2. *The card contains a code* $\mathsf{Pay.Code}$ *to be introduced in the smartphone app.*

Protocol 4. ***Group set-up protocol.***

1. *Some user* $\mathbf{id}^*$ *in the group of users* $\{\mathbf{id}_1, \ldots, \mathbf{id}_t\}$ *who want to use the service takes the leading role.* $\mathbf{id}^*$ *will be responsible for communicating with the verifying device.* $\mathbf{id}^*$ *sets his smartphone application to run in master mode and the other users in the group set their smartphones to run as slaves.*

2. *The users agree on an index* $j \in \{1, 2, \ldots, \ell\}$ *such that the value of the* j*-th identity-based public key in their respective vectors* $\mathbf{IK_{id_1}}, \ldots, \mathbf{IK_{id_t}}$ *is different for every user. Let these* t *different public keys be* $\mathsf{ik}_j^{d_j^1}, \ldots, \mathsf{ik}_j^{d_j^t}$.

Protocol 5. ***Group size accreditation protocol.***

1. *The master user* $\mathbf{id}^*$*'s device detects some verifying device and establishes a secure communication channel (the system may require the verifying device to authenticate to the user).*

2. *The verifying device sends to the master user* $\mathbf{id}^*$ *a unique time-stamped ticket* T *that may include a description of the service conditions and options.*

3. *The master user* $\mathbf{id}^*$ *distributes the ticket* T *along with the group parameters, namely the policy* Γ *(decided in the group set-up protocol) to the group members.*

4. *Each user* $\mathbf{id}_i$ *runs Algorithm* $\mathsf{IBDT.Sign}$ *to compute a partial signature with* Π *under his secret key* $\mathsf{sk}_j^{d_j^i}$ *on message*

$$\mathsf{Msg} = \left\langle \mathsf{T} \,||\, \mathsf{ik}_j^{d_j^1} \,||\, \ldots \,||\, \mathsf{ik}_j^{d_j^t} \right\rangle,$$

   for the threshold predicate $\Gamma = (t, \{\mathsf{ik}_j^{d_j^1}, \ldots, \mathsf{ik}_j^{d_j^t}\})$. $\mathbf{id}_i$ *sends the resulting partial signature* $\sigma_{\mathbf{i}}$ *to* $\mathbf{id}^*$.

5. *After receiving* $(\sigma_1, \ldots, \sigma_t)$, $\mathbf{id}^*$ *runs Algorithm* $\mathsf{IBDT.Comb}$ *to combine these partial signatures and output a final signature* σ *on behalf of* $\mathbf{id}_1, \ldots, \mathbf{id}_t$. $\mathbf{id}^*$ *sends*

$$\mathsf{Msg}' = \langle \mathsf{Msg}, \sigma \rangle$$

   to the verifying device.

6. *The verifying device checks the signature validity by running*

$$\mathsf{IBDT.Verify}(\mathsf{Msg}, \sigma, \mathsf{ik}_j^{d_j^1} || \ldots || \mathsf{ik}_j^{d_j^t}, t).$$

   Note that this signature will only be valid if all users $\mathbf{id}_1, \ldots, \mathbf{id}_t$ *have collaborated to compute it, and thus it proves that the group consists of at least* t *users. If the signature turns out to be invalid, the group will be punished in a way dependent on the particular application,* e.g. *by being denied access, being denied a group discount, etc. Otherwise, the service provider serves the group of users and tells the group the amount* amount_t *they have to pay depending on the group size.*

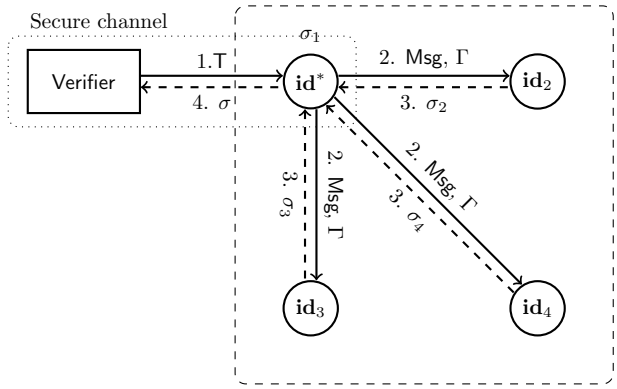

Figure 3.1: Group size accreditation protocol

Protocol 6. ***Payment.***

1. *Each group member* $\mathbf{id}$ *in the (sub)set* P *of those who are willing to contribute paying the bill sends via Bluetooth to the verifying device his payment code encrypted under the public key of SP:*

$$C_{\mathbf{id}} = \mathsf{Enc}_{\mathsf{pk}^{SP}}(\mathsf{T}||\mathit{Pay.Code}_{\mathbf{id}}),$$

where $\mathit{Pay.Code}_{\mathbf{id}}$ *is the code that* $\mathbf{id}$ *obtained from his prepaid scratch card and* Enc *is the public-key encryption algorithm of scheme* Π'.

2. *The verifying device obtains the payment codes of the users who will pay by decrypting the ciphertexts* $\{C_{\mathbf{id}} : \mathbf{id} \in P\}$*; then, the verifying device deducts the amount* amount_t *divided by* $|P|$ *(number of users who are collaborating in the payment) from the accounts associated with the received payment codes.*

3.6 Security and privacy

Our mechanism offers security and privacy by design:

- As stated by Theorem 1, the chosen IBDT scheme is selective-predicate and adaptive-message unforgeable. In plain words, for any $t \geq 2$, no group of less than t buyers can cheat the service provider by producing a threshold signature with threshold t.

- During the protocol execution, the service provider learns only the pseudonyms and the number of group members. Buyers preserve their anonymity *within those buyers that share the same public key* by virtue of the key management scheme described in Section 3.4. For instance, if each public key is associated with a combination of η decimal digits of the users' unique identifiers, then on average this public key is shared by a fraction $10^{-\eta} \times 100\%$ of the total number of users.
- By using a fully anonymous payment system, the anonymity level achieved by key management, whatever it is, is preserved after payment. In our case, payment anonymity is ensured by preventing Pay.Code from being linkable to any specific buyer. This can be achieved, for example, if the scratch card containing the Pay.Code is purchased using cash.

On checking physical closeness

In some applications, the verifier wants to check not only that the group consists of t or more members, but also that the members are physically close. For example, this is the case in HOV toll discounts, where all group members should be in the same car. The most viable solutions to check proximity are technology-based. One option is for the verifying device to check that at least t user devices (say smartphones) are Bluetooth-visible within a, say, 5 m range (note that in general seeing Bluetooth identifiers does not leak the identities of the device owners). Alternatively (or in addition), if trust can be placed on the app running the protocol in each user's smartphone, one can rely on the app checking that it is actually running in a real smartphone (this is actually checked by most apps) and that the smartphone is located near the verifying device (e.g. because it sees the verifying device). Other, more sophisticated security measures can be imagined to prove nearness, but the typical amount that can be earned by cheating the system (group discounts at tolls or museums) is too small to require that security level.

On cheating with multiple devices per user

If one or more group members carry multiple devices, the group could cheat by proving a size greater than its actual size. However, as said above, the typical amount that is at stake in a transaction of our protocol is small (a group discount, for example). Hence, maintaining and regularly carrying several smartphones to win that amount is probably not worth it. Also, if the registration process and key generation are based on the user's national ID card or social

security number (rather than on the phone number), a user cannot obtain more than one set of keys.

3.7 Performance analysis

Our proposed mechanism is designed to be executed by heterogeneous devices, such as servers, dedicated verifiers and users' smartphones. For this reason, and keeping in mind that the users' devices may be limited in computing power and often reliant on batteries, it is important that the computations of the underlying *IBDT* signature scheme be as fast as possible.

In this section, we provide a theoretical analysis of the complexity of the underlying *IBDT* signature scheme. We assess complexity by counting the number of point multiplications, point exponentiations and pairings, which are the costliest operations.

Table 3.1 shows the number of operations for each algorithm in the *IBDT* signature scheme. Operation counts are given in terms of the maximum number n of possible signature participants and the size t of the signing group. Like said above, $t \leq n$.

Table 3.1: Operations required per algorithm. **Mul** stands for multiplications and **Exp** for exponentiations.

	Mul $\mathbb{G}_1$	**Mul** $\mathbb{G}_2$	**Exp** $\mathbb{G}_1$	**Exp** $\mathbb{G}_2$	**Pairings**
Setup	0	0	$n+5$	$n+4$	1
Keygen	n^2+n	0	n^2+3n	0	0
Sign	$2n+7$	0	$2n+6$	0	0
Comb	$n^2+(3-t)n+2$	0	$n^2+(2-t)n+t$	0	0
Verify	0	$n+3$	0	$n+2$	3

Admittedly, the Sign and Comb algorithms, which are to be run in the users' smartphones, seem to require a high number of operations. This is certainly a potential drawback, because Sign and Comb need to be executed in every group accreditation attempt by the user's smartphones, whose computing power and energy are likely to be limited. Besides, lengthy smartphone computations may be outright unaffordable in some application settings: take for example a very

busy toll road, where toll stations become congested if takes too long for cars to pay. In contrast, the amount of computation is not a critical issue for the rest of the algorithms, since they will be run less frequently, or with no real-time constraints. Moreover, Setup, Keygen and Verify are typically run on devices with more resources than the users' devices.

We propose to precompute as many parts of the Sign and Comb algorithms as possible, and provide alternative descriptions for both of these algorithms.

The Sign algorithm is a probabilistic protocol; hence, not all of its operations can be precomputed. However, most operations depend on static values, *e.g.* keys and threshold policies Γ. Threshold policies contain the number of signers that will participate in a signature and their public keys. If we can assume that groups of users are relatively stable, *i.e.* if users generally use services together with the same group members or at least with a limited set of different groups, we can precompute operations that only depend on static values and threshold policies. Examples of values that could be precomputed are those in Expressions (3.2) and (3.3), among others.

The Comb algorithm depends on the output of the execution of Sign by all group members, but it is a deterministic algorithm, and most of its operations depend on the user's private key, the master public key, the public parameters, or the threshold policies, which are all known in advance. Therefore, by the same assumption as before, we can precompute some of the operations.

The precomputation approach splits Sign and Comb into two phases each, one for precomputing values, that can be executed during the group set-up protocol (Protocol 4), and a second phase executed during the group size accreditation protocol (Protocol 5) itself. The complexity of the resulting algorithms is presented in Table 3.2, where SignPrecomputation and CombPrecomputation are the respective precomputation phases, and FastSign and FastComb are the respective interactive phases run during Protocol 5.

3.8 Experimental results

We have written a Java library implementing the *IBDT* signature scheme to obtain experimental execution times. We have also implemented the entire accreditation method and tested it in a real scenario.

Performance tests have been run on an Ubuntu 15.04 x64 system with an Intel Core i7-3517U @ 1.90GHz and 8GB of DDR3 memory @ 1600MHz. The signature scheme has been implemented in Java7, using the java-7-openjdk-i386 environment and the java Pairing-Based Cryptography library [36]. Our

Table 3.2: Splitting of operations when precomputing the Sign and Comb algorithms. **Mul** stands for multiplications and **Exp** for exponentiations.

	Mul $\mathbb{G}_1$	**Exp** $\mathbb{G}_1$	**Pairings**
SignPrecomputation	$2n+3$	$2n+1$	0
FastSign	4	5	0
CombPrecomputation	$n^2+(3-t)n-3t$	$n^2+(2-t)n-2t$	0
FastComb	$3t+2$	$3t$	0

choice of elliptic curve was a Type F curve, with $|r| = 160$ bits, which makes elements in $\mathbb{Z}_q^\star$ 160 bits long, elements in $\mathbb{G}_1$ 320 bits long and elements in $\mathbb{G}_2$ 640 bits long. This should be enough to defeat attacks to the discrete logarithm problem, while keeping keys as short and operations as efficient as possible. The description of curve types and recommendations on how to choose curves and related parameters can be found in [69].

The rest of this section shows and discusses the execution times of our accreditation method, considering the times obtained with and without precomputation. We use the same notation for each of the algorithms as in the previous section, dividing the Sign and Comb algorithms into two-phase algorithms (SignPrecomputation, FastSign) and (CombPrecomputation, FastComb), respectively. Moreover, the protocol has been tested for multiple values of n and t.

Execution times of the algorithms, without precomputation, are shown in Figure 3.2. Setup times range between 650 and 1100 milliseconds, increasing linearly with n, as expected from the theoretical analysis conducted in the previous section. The value of t has no effect in the execution time of this algorithm. The generation of the curve parameters takes a constant time of approximately 300 ms. The Keygen algorithm is again independent from the value of t and grows polynomially with n. This is the expected behavior, as the size of the signing group does not influence the key generation procedure. Actually, this is one of the advantages of using a dynamic threshold scheme. The Sign algorithm shows again a behavior consistent with the theoretical analysis performed in the previous section, and does not depend on the size t of the group. This may sound counterintuitive, but the group size affects the Comb algorithm, rather than Sign. Finally, the Verify algorithm execution times are highly dominated by the 3 pairing operations it has to compute.

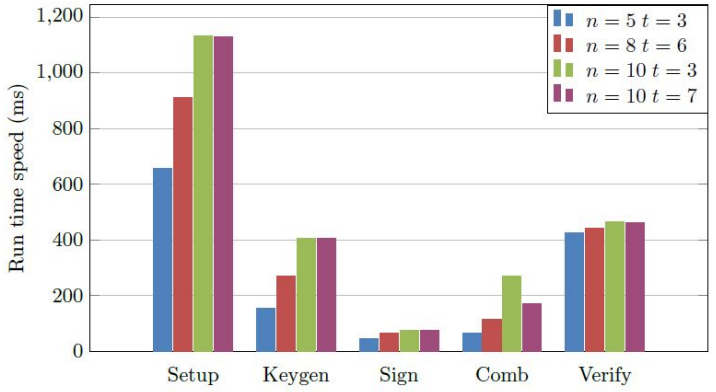

Figure 3.2: Execution times of the protocol without precomputation for different values of n and t

Figure 3.3 shows the execution times of the Sign and Comb algorithms when precomputation is performed. Thus, these algorithms are shown split as (Sign-Precomputation, FastSign) and (CombPrecomputation, FastComb).

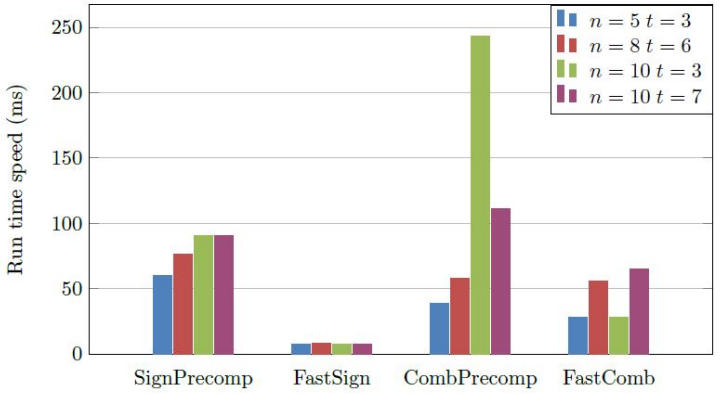

Figure 3.3: Execution times of the protocol with precomputation for different values of n and t

It is worth noting that the sum of the execution times of SignPrecomputation

and FastSign is very similar to the execution time of Sign. This shows that performance is at least as good as when not splitting operations. Moreover, the FastSign algorithm has a constant and very small execution time of around 10 ms. This demonstrates that the addition of precomputation phases is very effective to speed up the interactive phase of our method. Although the execution times of SignPrecomputation are a bit higher, this is of no concern, because this algorithm can be executed off-line and/or as a background process when the group is formed.

In the case of the CombPrecomputation and FastComb, we see that the precomputation phase is not as beneficial. Although the total execution time is still the same as the one of the original Comb algorithm, and thus precomputation does not penalize performance, the execution times are more evenly divided between the precomputation phase and the interactive phase. This could somehow be expected, as FastComb has more dependencies than FastSign: as predicted in Section 3.7 and Table 3.2, it depends on t (the larger t, the longer it takes). On the other hand, CombPrecomputation depends on $n - t$, rather than separately on n and t; the larger $n - t$, the longer CombPrecomputation takes. The above dependencies are further explored in Figure 3.4.

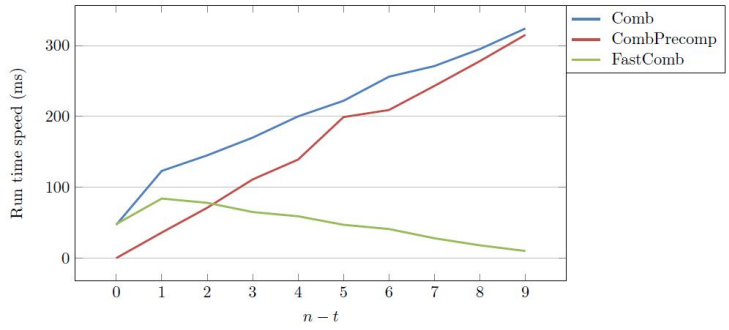

Figure 3.4: Comb times as a function of $(n - t)$

3.9 High-occupancy vehicles use case

In order to assess the applicability of our proposed mechanism, we carried out a pilot experiment related to HOVs. Our pilot application allows HOVs to

find and use parking spots especially designated for them and get reduced fees depending on the number of car passengers. These parking spots for HOVs are located in parking lots guarded by an automatic barrier.

The pilot is composed of:

- a passenger Android application to be run by the smartphone of each car passenger with the functionalities described in Section 3.5;
- a verifier application that runs in the automatic barrier of the parking lot.

We take the telephone number as the passenger identifier and we set the protocol parameters to $\eta = 1$ and $n = 5$, as we want to accredit groups ranging from 2 to 5 passengers in each car. The communication between passenger applications, needed to compute IBDT signatures, is performed using NFC while the communication between the leading passenger application and the verifier application running in the barrier is performed using Bluetooth only.

Figure 3.9 shows screenshots taken from the passenger application. In the left-hand side screenshot, the application shows to the passenger where free HOV parking spots can be found within a nearby parking lot. In the central screenshot, the leading passenger application chooses a slot and starts the group size accreditation protocol at the parking lot barrier. In the right-hand side screenshot, after combining partial signatures of two passengers (Alberto and Test), a group size of two passengers is accredited to the barrier verifier.

The pilot includes access control to the parking lot and the accreditation mechanism. However, while it calculates the parking fee according to the number of passengers, at the moment no specific payment system is embedded in the pilot. Payment will be incorporated in case of commercial deployment.

3.10 Summary

We have presented a mechanism to accredit the size of a group (for example, in order to obtain a group discount) that is compatible with the anonymity of its members and with dynamic group formation. To the best of our knowledge, this is the first mechanism that allows cryptographically proving the number of members of a group in an anonymous way and without the need of dedicated hardware. Our protocol suite is built upon a new cryptographic primitive called *IBDT* signature scheme, a novel key generation and management solution, short-range communication technologies and, in case payment is needed,

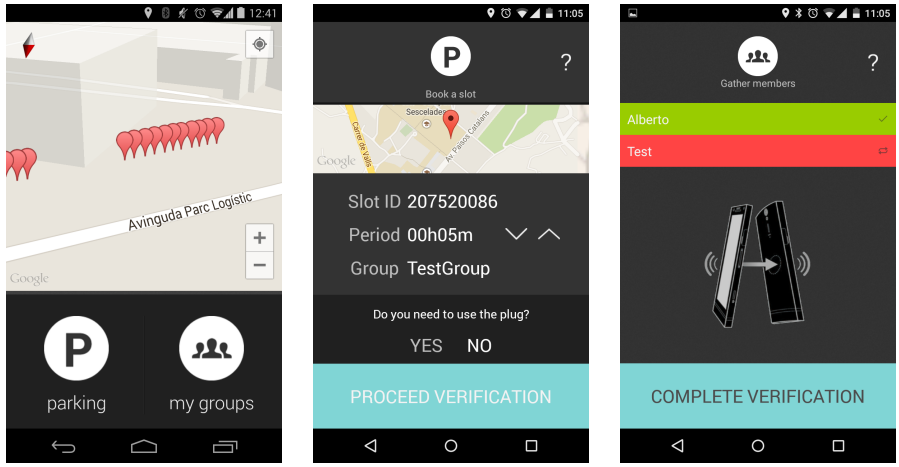

Figure 3.5: HOV pilot passenger application. Left, locating free HOV spaces. Center, booking a space and starting group size accreditation. Right, completing accreditation of a group of two passengers.

anonymous payment mechanisms. The reported complexity analysis and simulations, as well as the pilot experiment we have deployed in a real scenario, show that our system is usable in practice.

Chapter 4

Loyalty Programs

4.1 Introduction

Any vendor is extremely interested in establishing lasting relationships with consumers. For some companies (like public utilities or banks) long relationships are the rule rather than the exception, whereas for other companies (like retailers) consumer loyalty is much harder to obtain without specific incentives. Loyalty programs are instruments whereby vendors try to provide such incentives. In a loyalty program, the vendor pursues two main goals: i) to encourage the consumer to make more purchases in the future (returning customer); ii) to allow the vendor to profile the consumer in view of conducting market research and segmentation (profiled customer). In order to lure consumers into a loyalty program, the vendor offers them rewards, typically loyalty points that consumers can later exchange for discounts, gifts or other benefits offered by the vendor. Normally, enrollment to loyalty programs involves some kind of registration procedure, in which customers fill out a form with their personal information and are granted a loyalty card, be it a physical card (magnetic stripe or smartcard) or a smartphone application.

Market analysis and customer segmentation are carried out by building profiles of individual customers based on their personal information, which customers supply to the vendor during enrollment to the loyalty program, and their purchase records, collected every time customers present their loyalty cards. The profiles thus assembled are used in marketing actions, such as market studies and targeted advertising.

Although loyalty programs have become widespread, they are experiencing a loss of active participants and they have been criticized by business experts and consumer associations. Criticism is mainly due to privacy issues, because it is not always clear whether the benefits vendors offer in their loyalty programs are worth the loss of consumer privacy caused by profiling [78, 91, 1, 45].

Loyalty programs can offer clear advantages to both vendors and consumers, like returning customers and special discounts, respectively. However, privacy concerns regarding buyer profiling affect more and more the acceptance of such programs, as the public awareness on the dangers of personal information disclosure is increasing.

4.1.1 Contributions

In this Chapter we propose a protocol for privacy-aware loyalty programs that allows vendors and consumers to enjoy the benefits of loyalty, while preserving the anonymity of consumers and empowering them to decide how accurately they reveal their profile to the vendor. In order to encourage customers not just to return but also to disclose more of their profile, the vendor must offer *additional* rewards to consumers. Thus, vendors *pay* consumers for their private information. On the other hand, consumers become aware of how much their personal data are worth to vendors, and they can decide to what extent they are ready to reveal such data in exchange for what benefits.

To empower consumers as described above, we provide them with a mechanism that allows them to profile themselves, generalize their profiles and submit these generalized profiles to the vendor in an anonymous way. There are some technical challenges to be overcome:

- The proposed mechanism should prevent vendors from linking the generalized profiles to the identity of buyers, to particular transactions or to particular loyalty points submitted for redemption.

- To prevent straightforward profiling by the vendor, payment should be anonymous. In online stores, to completely achieve anonymity, the buyers should use some kind of anonymous payment system, such as Bitcoin [72], Zerocoin [71], some other form of electronic cash [32], or simply scratch cards with prepaid credit anonymously bought, say, at a newsstand. In physical stores, it would be enough to pay with cash.

- Consumers should not be able to leverage their anonymity to reveal forged profiles to the vendor, which would earn them rewards without actually

revealing anything on their real purchase pattern.

Our proposed mechanism, thus, needs to take care of the two main aspects of loyalty programs. First, it has to provide a way to obtain and submit loyalty points in an anonymous and unlinkable way; that is, a customer should be able to submit a particular loyalty point to a vendor, but the vendor should not be able to link that particular loyalty point to the transaction in which it was issued. Second, our mechanism must allow customers to build their own generalized profiles from their respective purchase histories, but it must prevent customers from forging false profiles and vendors from linking the generalized profiles to particular customers. We will show later that these two aspects can be tackled in a similar way.

The contributions in this Chapter have been published in [13, 14].

The Chapter is organized as follows. Section 4.2 briefly recalls past works on this subject. Section 4.3 starts recalling the functionalities of a conventional loyalty program; then it formalizes the notion of privacy-aware loyalty program; after that, it goes on to present the security and functionality requirements that our new privacy-aware protocol suite should satisfy. In Section 4.4, we explain how we use generalization of purchase receipts while preventing the existence of several generalizations of each purchase receipt from being abused to submit the receipt more than once to get loyalty points. In Section 4.5 we introduce anonymous tokens with controlled linkability based on partially blind signatures. In Section 4.6 we present our privacy-aware loyalty program protocol suite, that builds on the tools described in Sections 4.4 and 4.5. In Section 4.7 we analyze the computational complexity and the security of the suite. In Section 4.8 we present experimental results, including an implementation on smartphones and an analysis of deployability in physical and online stores. Section 4.9 presents an extension of our construction that guarantees untransferability of purchase receipts and/or loyalty points. Finally, Section 4.10 summarizes conclusions.

4.2 Related work on loyalty programs

Our method aims to offer all the functionalities of loyalty programs; that is, to allow vendors to reward returning customers with loyalty points and to profile returning customers based on their purchase histories. The novelty is that our scheme empowers customers with the ability to decide how accurately they disclose their purchase histories to vendors.

A simple and perhaps the most widespread approach to implement a loyalty program is to have a centralized server, owned and operated by some vendor

$\mathcal{V}$, that stores the information on the program participants. This information includes all the personal data the participants gave to the vendor when they enrolled to the program, their balance of loyalty points, and their history of purchases. Each customer is given a loyalty card which contains the identifier of her record in the server's database. Each time a customer buys at a store and presents her loyalty card, her record in the server is updated, by adding to it the items she bought and modifying her balance of loyalty points if needed. In this way, all transactions by each customer can be linked to each other using the customer's identifier. Even if the customer provided false information when she enrolled to the loyalty program, all of her transactions would be linked anyway. Hence, discovering the customer's identity in one individual transaction (*e.g.* through the credit or debit card used for payment) would allow linking her entire profile to her real identity.

If control over profiling and purchase histories is to be left to customers, a centralized approach does not seem a good solution. Moreover, we should also ensure that individual transactions cannot be linked to each other unless desired by the customer. To do so, we will let each customer manage locally and anonymously her own balance of loyalty points and history of purchases.

To the best of our knowledge, no work on privacy-preserving loyalty programs other than ours has been published to this date. However, if we focus on privacy-preserving customer profiling, there are certainly other works in which a service provider collects information on her user base in a privacy-preserving way. For example, the authors from [79] propose a scheme to collect location-based aggregate statistics. In that contribution, drivers provide location-based information (*e.g.* their speed), to a centralized server that can aggregate this information. The information they provide is always encrypted, using some homomorphic encryption scheme. The data they deal with is only numeric, and the computations that can be done are defined beforehand (they depend on the homomorphic encryption scheme in use). In our contribution, however, data do not need to be numeric, and we protect them by generalization rather than encryption; note that, unlike homomorphic encryption, generalization does not restrict computations that can be carried out on the protected data.

4.3 Definition of privacy-preserving loyalty programs

Our proposed mechanism follows the decentralized approach. To allow local management of loyalty points and purchase receipts by the customer, we treat points and receipts as anonymous electronic cash, in the sense that: i) they are one-time certified tokens of information; ii) they are issued by vendors and they can only be redeemed at the same vendor who issued them, but issued tokens and redeemed tokens should remain unlinkable. However, unlike in anonymous electronic cash schemes, in our scheme the entity issuing certified tokens of information is not a trusted third party: indeed, the issuer in our scheme is the vendor, and placing complete trust in the vendor would allow him to profile the users. Moreover, the concrete implementation of the loyalty program should discourage customers from transferring loyalty points and purchase receipts among them. Purchase histories will be built by the vendor from the individual purchase receipts of all products purchased by each customer *that the customer allows the vendor to link together*; furthermore, the customer can decide how generalized/coarsened are the product descriptions in the purchase receipts she allows the vendor to link to one another.

Definition 14 (Privacy-aware Loyalty Program)**.** *A Privacy-aware Loyalty Program scheme has three participants: a key dealer or certification authority $\mathcal{CA}$, a vendor $\mathcal{V}$, and a customer $\mathcal{C}$. $\mathcal{V}$ keeps a set $\mathcal{DB}$ of submitted tokens that is initially empty. The scheme consists of seven protocols (*Setup, VendorSetup, Enroll, Buy, Submit, Issue, Redeem*):*

- Setup *is a probabilistic polynomial-time algorithm run by $\mathcal{CA}$ in which, on input a security parameter γ, outputs (and publishes) the system parameters* params*.*
- VendorSetup *is a probabilistic polynomial-time key generation algorithm. $\mathcal{V}$ is certified as a legitimate vendor and obtains* params *and a key pair $(pk_{\mathcal{V}}, sk_{\mathcal{V}})$ from $\mathcal{CA}$.*
- Enroll *is a protocol run by $\mathcal{C}$ whereby $\mathcal{C}$ obtains access to the loyalty program, typically by registering and obtaining* params *and $pk_{\mathcal{V}}$.*
- Buy *is a probabilistic polynomial-time interactive protocol run between $\mathcal{V}$ and $\mathcal{C}$. The public inputs of both $\mathcal{C}$ and $\mathcal{V}$ contain a product name p. The*

private input of $\mathcal{C}$ contains a message y, and that of $\mathcal{V}$ contains the private key $sk_{\mathcal{V}}$. When the protocol finishes, the private output of $\mathcal{C}$ contains either `fail` *or a list of receipt tokens $(R_i)_{i=1,\ldots,n}$.*

- Submit *is a polynomial-time algorithm that takes $pk_{\mathcal{V}}$, a receipt token R and $\mathcal{DB}$ as inputs.* Submit *outputs either* `accept` *if the signature on the token R is valid and $R \notin \mathcal{DB}$ or* `reject` *otherwise. An accepted submitted token is thereafter invalidated by adding it to $\mathcal{DB}$.*

- Issue *is a probabilistic polynomial-time interactive protocol run between $\mathcal{V}$ and $\mathcal{C}$. The public inputs of both $\mathcal{C}$ and $\mathcal{V}$ contain a value c of loyalty points. The private input of $\mathcal{C}$ contains a message y, and that of $\mathcal{V}$ contains a private key $sk_{\mathcal{V}}$. When the protocol finishes, the private output of $\mathcal{C}$ contains either* `fail` *or a loyalty points token P.*

- Redeem *is a polynomial-time algorithm that takes $pk_{\mathcal{V}}$ and loyalty points token P and $\mathcal{DB}$ as inputs.* Redeem *outputs either* `accept` *if the signature on the token P is valid and $P \notin \mathcal{DB}$ or* `reject` *otherwise. A redeemed token is then invalidated by adding it to $\mathcal{DB}$.*

4.3.1 Requirements

Definition 15 (Correctness and security)**.** *A Privacy-aware Loyalty Program is considered correct and secure if, given* params $\leftarrow$ Setup(γ), $(pk_{\mathcal{V}}, sk_{\mathcal{V}}) \leftarrow$ VendorSetup(*params*)*, it fulfills the following properties:*

- *(Correctness) For any product name p, any private input y of $\mathcal{C}$, a private key $sk_{\mathcal{V}}$, and a receipt token $R \leftarrow$ Buy$(p, y, sk_{\mathcal{V}})$ such that $R \notin \mathcal{DB}$, the execution of* Submit$(pk_{\mathcal{V}}, R, \mathcal{DB})$ *must return* `accept` *with overwhelming probability. Likewise, for any value c, any private input y of $\mathcal{C}$, a private key $sk_{\mathcal{V}}$, and a loyalty points token $P \leftarrow Issue(c, y, sk_{\mathcal{V}})$ such that $P \notin \mathcal{DB}$, the execution of $Redeem(pk_{\mathcal{V}}, P, \mathcal{DB})$ must return* `accept` *with overwhelming probabiliy.*

- *(Unforgeability) Receipt tokens R and loyalty points tokens P should be unforgeable against one-more forgery under chosen-message attacks; that is, for any integer ℓ, no polynomial adversary $\mathcal{A}$ should be able to successfully submit $\ell+1$ receipt tokens after only ℓ executions of the* Buy *protocol, nor redeem $\ell+1$ loyalty points tokens after only ℓ executions of the* Issue *protocol.*

- *(Anonymity) A vendor $\mathcal{V}$ should not be able to link a submitted receipt token R to the particular execution of the* Buy *protocol in which the token was produced. Likewise, $\mathcal{V}$ should not be able to link a submitted loyalty points token P to the particular execution of the* Issue *protocol in which the token was produced.*

Beyond the above generic security properties, the proposed scheme should provide the following two security properties related to token management:

- **Controlled linkability**. A customer $\mathcal{C}$ should be able to decide whether a submitted receipt token R can be linked to other receipt tokens submitted by the same $\mathcal{C}$ to the same vendor $\mathcal{V}$.
- **Untransferability**. A customer $\mathcal{C}$ should not be able to obtain loyalty points from $\mathcal{V}$ by submitting a purchase receipt issued to another customer $\mathcal{C}'$; transfer of purchase receipts among customers blurs their profiles and hence is against the vendor's interests. Note that transfer of purchase receipts among customers implies that the transferring customer loses the loyalty points she could get in exchange for that receipt, as submitted receipt tokens are invalidated. On top of that, the customer who transfers her receipt tokens loses some privacy w.r.t the customer to whom she transfers them. If losing loyalty points and partially losing one's privacy can be assumed sufficient to discourage the transfer of receipts, no specific countermeasures are needed to ensure untransferability. For the case where the above assumption does not hold, in Section 4.9 we provide an extension of our protocol suite to enforce untransferability of both receipt and loyalty points tokens.

4.4 Generalization of purchase histories

In our protocol, we allow the buyer to choose the level of generalization when disclosing her purchase history to claim loyalty points; in this way, the buyer is put in control of her privacy. To that end, the vendor provides the buyer with receipts for all possible generalizations of each product purchased by the buyer. However, the protocol must be designed in such a way that the buyer cannot cheat by using different generalized receipts corresponding to the same purchased product as if they were receipts of different purchased products, in order to obtain more loyalty points.

To generalize receipts, a vendor must use a publicly available taxonomy for the products she offers. This taxonomy $\mathcal{T}$ is modeled as a tree, being its root

node a generic identifier such as *Product*, and each leaf a specific product in the set of products $P = \{p_1, \ldots, p_n\}$ on sale. The inner nodes of the tree are the subsequent categories to which the products belong: the closer to the leaf nodes, the more specific categories are.

A generalization function $g : \mathcal{T} \to \mathcal{T}$ returns the parent of a node. Applying the generalization function m times will be denoted as g^m. As an example, for the product $p_i =$ "$Inception$", its generalizations might be $g(p_i) = ActionMovie$, $g^2(p_i) = Movie$, $g^3(p_i) = DigitalMedia$ and $g^4(p_i) = Product$. For simplicity and ease of implementation, it is desirable that all leaves be at the same depth, that is, that the path from the root to any leaf be of the same length.

Customers in our loyalty program protocol will receive a list of purchase receipts $(R_1, \ldots, R_n)$ for every product they purchase. This list contains a receipt for the specific product and receipts for all of its generalizations in the path up to the root of the taxonomy (generalization path).

Now, when a customer decides to submit her purchase history, she chooses how much she want to generalize each purchase in her history, from no generalization (the actual name of the product) to maximum generalization (just the top category *Product*). Then the customer is required to send for each purchase *all* the tokens in the purchase generalization path from the chosen generalization level up to the root of the taxonomy. Following the movie example above, a customer who wants to submit her purchase generalized to level 2 will submit the tokens $Movie$, $DigitalMedia$ and $Product$.

Forcing customers to send all tokens from the selected generalization level to the root prevents them from using tokens in the generalization path of a purchase to falsely claim additional purchases.

4.5 Anonymous tokens with controlled linkability

Loyalty points and purchase receipts have requirements in line with those of anonymous electronic cash and anonymous electronic credentials. These well-known primitives commonly use blind signatures and/or zero-knowledge proofs of knowledge [28, 29, 32, 73]. We will treat loyalty points and purchase receipts using a construction that we call anonymous tokens with controlled linkability. These tokens will be realized by using a partially blind signature construction, namely the one described in [98], with slight changes introduced in the messages

to be signed.

4.5.1 Controlled linkability of tokens

The use of partially blind signatures will ensure that a submitted token cannot be linked to an issued token, nor to the customer to whom it was issued. However, if vendors are to be allowed to build customer profiles from anonymous purchase receipts, there must be a mechanism whereby, if allowed by the customer, the vendor can verify that several submitted purchase receipt tokens really correspond to the same (anonymous) customer, even if receipts have been generalized by the customer prior to submission. Note that if all (ungeneralized) purchase receipts from the same customer could be linked, customer anonymity would be problematic in spite of partially blind signatures: a very long and detailed profile is likely to be unique and goes a long way towards leaking the customer's identity.

Thus, we propose a mechanism that allows customers to decide which purchase receipt tokens can be linked together, by employing an additional identifier as part of the secret message in the partially blind signature. This identifier is chosen by the customer for each receipt token at the moment of token issuance. If a customer picks a fresh random number for each issued purchase receipt, then none of this customer's receipts will be linkable to each other; however, if the customer uses the same identifier for a group of purchase receipt tokens at the time of token issuance, then all of the tokens in this group can be verifiably linked together by the vendor after they are submitted.

4.5.2 Description

We use a partially blind signature scheme from bilinear pairings presented in [98]. This scheme satisfies the requirements of correctness, partial blindness and unforgeability against one-more forgery under chosen-message attacks in the random oracle model under the inverse CDH assumption in bilinear groups, and thus it is considered secure. Security proofs can be found in [98]. Additionally, this scheme produces short signatures, it is computationally efficient and allows aggregate verification of signed messages bearing the same agreed public information. Note that we use this partially blind signature scheme as a black box, and any other secure scheme would fit in our proposed protocol. For completeness, we describe this scheme below.

Anonymous tokens with controlled linkability are operated in four phases:

- In the *setup* phase, a certification agency generates the public parameters of the partially blind signature scheme.
- In the *key generation* phase, users (*i.e.* vendors and customers) get their key pairs from the certification agency.
- In the *issuance* phase, a token corresponding to some loyalty points or to a purchase receipt is generated by a customer, it is signed in a partially blind way by a vendor and it is returned to the customer.
- Finally, in the *verification* phase, a customer submits previously generated tokens to a vendor, who in turn verifies that each token was correctly signed. If tokens correspond to purchase receipts, the vendor may verify whether the submitted tokens are linked with each other and/or with previously submitted tokens.

Setup

This algorithm is executed once by a certification authority to set up the system parameters. It takes as input a security parameter γ. The algorithm chooses bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of order $q > 2^\gamma$, an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, a generator $g \in \mathbb{G}_1$ and collision-resistant hash functions $H : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_0 : \{0,1\}^* \to \mathbb{G}_2$. The public parameters are $\mathsf{params} = \{q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, H, H_0\}$.

Key generation

A vendor gets a secret key $sk_{\mathcal{V}} = x \in_R \mathbb{Z}_q^*$ and a public key $pk_{\mathcal{V}} = g^x$, and publishes his public key.

Token issuance

A customer wants to obtain from a vendor a token with an agreed public information c (this information may specify a number of loyalty points or a purchase receipt for a certain product). This is an interactive protocol which produces a partially blind signature on public information c, and a secret message containing a unique identifier α of the token and a (possibly) unique identifier y. The protocol is depicted in Figure 4.1 and described next:

1. The customer chooses a value for y, either from a list of previously used values or by generating a new one uniformly at random from $\mathbb{Z}_q^*$.

2. The customer and the vendor agree on a public string $c \in \{0,1\}^*$.

3. The customer chooses random $\alpha, r \in_R \mathbb{Z}_q^*$ and builds the message $m = (\alpha, y)$. Then, the customer blinds the message by computing $u = H_0(c||m)^r$ and sends u to the vendor.

4. The vendor signs the blinded message by computing $v = u^{(H(c)+sk_{\mathcal{V}})^{-1}}$ and sends it back to the customer.

5. The customer unblinds the signature by computing $\sigma = v^{r^{-1}}$. The resulting tuple $T = \langle c, m, \sigma \rangle$ is the token.

An execution of this protocol, between a vendor $\mathcal{V}$ and a customer $\mathcal{C}$, is denoted by $T = \langle c, m, \sigma \rangle = \mathsf{Issuance}(\mathcal{V}, \mathcal{C}, c, y)$.

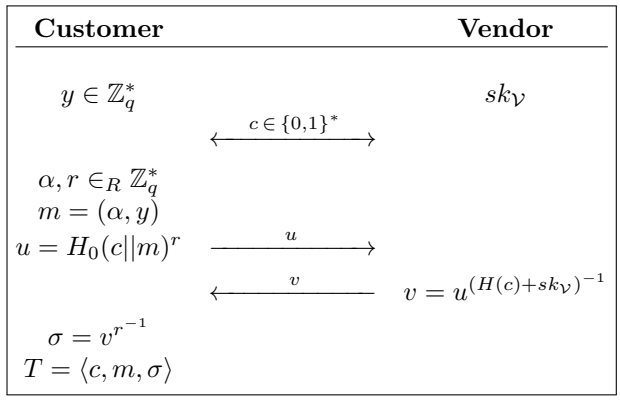

Figure 4.1: Issuance protocol

Token verification

The submission and verification of a token is an interactive protocol between a customer and a vendor. The customer submits the token $T = \langle c, m, \sigma \rangle$ and the vendor returns `accept` or `reject` as a result of the verification. Informally, the vendor checks that the signature on the token is valid and has been produced by himself; then, if the value y contained in the message matches the one of a previously submitted token, the tokens are grouped. The protocol is outlined in Figure 4.2 and described next:

1. The customer sends $T = \langle c, m, \sigma \rangle$ to the vendor.
2. The vendor parses the message m as (α, y).
3. The vendor verifies the signature by checking the equality

$$e(g^{H(c)} \cdot pk_{\mathcal{V}}, \sigma) \stackrel{?}{=} e(g, H_0(c||m)).$$

   If the above equality holds, he checks whether the token has already been spent (he verifies whether a token with the same α has previously been submitted). If the verification was successful and the token has not been spent yet, he marks it as spent and sends an `accept` message to the customer. Otherwise, he sends a `reject` message to the customer.
4. Finally, the vendor checks whether the identifier value y is the same as the one in a previously spent token. If yes, he links the new token with that previous one.

An execution of this protocol involving a customer $\mathcal{C}$, a vendor $\mathcal{V}$ and a token T is denoted as $\mathsf{accept/reject} = \mathsf{Verification}(\mathcal{V}, \mathcal{C}, T)$.

Customer		**Vendor**
T		$pk_{\mathcal{V}}$
	$\xrightarrow{T=\langle c,m,\sigma\rangle}$	
		Parse m as (α, y)
		$e(g^{H(c)} pk_{\mathcal{V}}, \sigma) \stackrel{?}{=} e(g, H_0(c\|\|m))$
	$\xleftarrow{\mathsf{accept/reject}}$	and token not spent yet.
		Link token to previous ones with same y

Figure 4.2: Verification protocol

Aggregate verification

This protocol allows the customer to aggregate signatures of messages bearing the same public information by just multiplying the resulting signatures. If there is a list of tokens $\{T_1, \ldots, T_n\}$, where $T_i = \langle c_i, m_i, \sigma_i \rangle$, and $c_i = c$ for $1 \leq i \leq n$, a customer can aggregate the partially blind signatures by computing

$\sigma_{agg} = \prod_{i=1}^{n} \sigma_i$ and submitting $T_{agg} = \langle c, \{m_1, \cdots, m_n\}, \sigma_{agg} \rangle$. The vendor can then verify the validity of the aggregated token by checking the equality

$$e(g^{H(c)} \cdot pk, \sigma_{agg}) \stackrel{?}{=} e(g, \prod H_0(c||m_i)).$$

4.6 Privacy-aware loyalty program construction

Our proposed solution for privacy-aware loyalty programs builds on the anonymous tokens with controlled linkability we described in Section 4.5 and the generalization of purchase histories described in Section 4.4. As introduced in Section 4.3, our construction consists of the following protocols: SETUP, VENDORSETUP, ENROLL, BUY, SUBMIT, ISSUE and REDEEM.

Protocol 1. ***Setup.***
The setup phase is run by a certification authority to generate the public parameters params *of the anonymous token with controlled linkability construction described in Section 4.5. These parameters are made public to every* $\mathcal{V}$ *offering loyalty programs and to every* $\mathcal{C}$ *intending to participate in them.*

Protocol 2. ***VendorSetup.***
Each vendor $\mathcal{V}$ *publishes a product taxonomy* $\mathcal{T}_{\mathcal{V}}$ *as described in Section 4.4. Then,* $\mathcal{V}$ *obtains a key pair built as described in the key generation procedure in Section 4.5. Finally,* $\mathcal{V}$ *publishes his public key.*

Protocol 3. ***Enroll.***
Customers obtain the public parameters params *and some means to communicate with the system, namely a smartcard or a smartphone application. Furthermore, customers enrolling to a loyalty program from a particular vendor obtain the vendor's public key and his taxonomy of products. This step is not mandatory, but it allows customers to check that tokens issued by vendors are valid and purchase receipt generalizations are correct.*

Protocol 4. ***Buy.***
A customer $\mathcal{C}$ *in a loyalty program offered by a vendor* $\mathcal{V}$ *purchases a product, either at a physical or online store of* $\mathcal{V}$*. Note that, in the case of an online store,* $\mathcal{C}$ *should use additional anonymization measures, such as anonymous Internet surfing, offered for example by Tor networks [3], anonymous shipping methods [59], and anonymous payment methods (*e.g. *[72, 32, 71] or simply prepaid scratch cards). The protocol is as follows:*

1. *$\mathcal{C}$ sends to $\mathcal{V}$ the name p_i of the product $\mathcal{C}$ wants to buy.*

2. *$\mathcal{C}$ chooses a value y to be used in the token issuance protocol, depending on her privacy preferences: if she wants the new purchase receipt to be linkable to previously obtained purchase receipts (linkability is incentivized as described in Section 4.6.1 below), she will reuse the same y that was used in those previous receipts; if she does not want this new purchase receipts to be linkable to previous receipts, she will pick a new random $y \in \mathbb{Z}_q^*$.*

3. *In order to produce purchase receipt tokens for product p_i and all its generalizations, $\mathcal{V}$ and $\mathcal{C}$ run the interactive protocol $\mathsf{Issuance}(\mathcal{V}, \mathcal{P}, p_i, y)$, $\mathsf{Issuance}(\mathcal{V}, \mathcal{P}, g(p_i), y)$, $\mathsf{Issuance}(\mathcal{V}, \mathcal{P}, g^2(p_i), y)$, etc. up to the root of the taxonomy. In this way, $\mathcal{C}$ obtains as many purchase receipt tokens as the depth of p_i in $\mathcal{V}$'s taxonomy.*

Protocol 5. ***Submit.***
At any moment, a customer can submit a list of purchase receipts (or a generalized version of them) to the vendor and obtain loyalty points. To this end, for each purchased product in her claimed purchase history, the customer sends the receipt token corresponding to the level of generalization she wishes. In particular, the customer could use maximum generalization *(a receipt just specifying that she has purchased a "product")* if she does not wish to disclose anything about what she has bought. In principle, the more generalization is used by the customer, the less loyalty points she can expect to be given by the vendor, although the correspondence between token detail and loyalty points depends on the vendor's reward policy.

Additionally, as said in Section 4.4, for each product she also submits all tokens from the selected generalization level up to the root of the taxonomy (to ensure tokens in the generalization path cannot later be used as independent purchase receipts). Submission of each token T_i is performed according to the $\mathsf{Verification}(\mathcal{V}, \mathcal{P}, R_i)$ protocol described in Section 4.5.2.

Protocol 6. ***Issue.***
To issue loyalty points, the vendor builds a message info *that encodes an identifier of the vendor, the number of points this token is worth and an expiration date. Unlike for purchase receipts, the vendor has no legitimate interest in linking several tokens containing loyalty points and thus does not incentivize linkability. Hence, the customer picks a fresh random y for each new loyalty points token she claims. Then the vendor and the customer run the interactive*

protocol $\mathsf{Issuance}(\mathcal{V}, \mathcal{P}, \mathsf{info}, y)$. *The generated token contains the loyalty points issued to the customer.*

To ensure that a loyalty points token submitted for redemption cannot be linked with an issued loyalty points token, the number of loyalty points associated to a single token should be limited to a small set of possible values, similar to the limited denominations of bank notes. There is an efficiency toll to be paid for this caution, as issuing a certain amount of loyalty points can require running the Issuance *protocol several times (several tokens may be needed to reach the required amount).*

Protocol 7. ***Redeem.***
A participant $\mathcal{C}$ who wants to redeem a loyalty points token T previously earned at a vendor $\mathcal{V}$'s in exchange for some benefits runs the interactive protocol $\mathsf{Verification}(\mathcal{V}, \mathcal{P}, P)$.

It is possible to simultaneously redeem several loyalty points tokens by using the aggregation of signatures described in Section 4.5.2.

4.6.1 Incentives related to purchase receipts submission

Vendors can establish strategies to incentivize or discourage certain customer behaviors:

- To encourage customers to use little or no purchase receipt generalization (and hence to renounce some of their privacy), the amount of loyalty points awarded per receipt token should depend on the chosen level of generalization: more loyalty points awarded to less generalized purchase receipts.
- If the customer submits unlinkable receipts, she should just get enough loyalty points to reward her as a returning customer. To encourage customers to allow linkage of purchase receipt tokens by the vendor (and hence customer profiling), a customer should get more loyalty points if she submits $n_1 + n_2$ tokens with the same y value than if she submits n_1 tokens with one y value and then n_2 tokens with a different y value (*superlinear reward*). Furthermore, the vendor may require that the list of linkable receipt tokens for which reward is claimed correspond to purchases made within a certain time window (if linking purchases very distant in time is uninteresting for profiling).
- Two or more customers might be tempted to share their y values in order to submit a longer list of linkable receipts and thereafter share the super-

linear number of loyalty points they would earn. As long the reward is only *slightly* superlinear, customer collusion is discouraged if the customer $\mathcal{C}$ who submits the list of linkable tokens is required by $\mathcal{V}$ to actually show all the actual linkable tokens (and not just a reference to them): colluders different from $\mathcal{C}$ may not like to pay the privacy toll of disclosing their purchase receipts to $\mathcal{C}$.

4.7 Complexity and security analysis

We count here the number of operations required by the Issuance and Verification protocols described in Section 4.5. These are the two performance-critical protocols, because they are the ones that need to be run every time a token (carrying a purchase receipt or loyalty points) is to be generated or verified.

The Issuance protocol requires computation by the vendor of 1 exponentiation in $\mathbb{G}_2$; also, 1 hash, 1 addition and 1 inversion in $\mathbb{Z}_q^*$. The customer computes 2 exponentiations in $\mathbb{G}_2$ and 1 inversion in $\mathbb{Z}_q^*$. The Verification protocol requires computation by the vendor of 1 exponentiation, 1 multiplication in $\mathbb{G}_1$ and 1 hash to $\mathbb{G}_2$; also, 1 hash in $\mathbb{Z}_q^*$ and 2 pairings.

As shown by the computing times given in Section 4.8 below, the above computations can be done in a very reasonable time, even though the customer's computation must be carried out by her smartphone.

Regarding security, we justify here that, the security requirements identified in Section 4.3.1 are satisfied by the above protocol suite.

- **Correctness.** If the partially blind signature is correctly computed during token issuance, the token verification equation will pass, because

$$\begin{aligned} e(g^{H(c)} \cdot pk, \sigma) &= e(g^{H(c)+x}, \sigma) \\ &= e(g^{H(c)+x}, v^{r^{-1}}) \\ &= e(g^{H(c)+x}, u^{(H(c)+x)^{-1} \cdot r^{-1}}) \\ &= e(g^{H(c)+x}, H_0(c||m)^{r \cdot r^{-1} \cdot (H(c)+x)^{-1}}) \\ &= e(g, H_0(c||m)^{r \cdot r^{-1} \cdot (H(c)+x) \cdot (H(c)+x)^{-1}}) \\ &= e(g, H_0(c||m)). \end{aligned}$$

- **Unforgeability.** Unforgeability against one-time forgery under chosen message attack of receipt and loyalty points tokens is provided by the partially blind signature scheme. Note that we use the partially blind

signature recalled in Section 2.4 as a black box: the Issuance and Verification protocols in Section 4.5.2 use partially blind signatures *exactly* as described in [98]; hence, the security proof of [98] guarantees the security of such protocols.

- **Anonymity.** No information on the user is obtained by a server during the protocol. Submitted tokens cannot be linked to issued tokens or to the identity of a requester or prover because of the *partial blindness* property of the signature scheme [98]. The justification of partial blindness given in [98] is briefly recalled below.

- **Controlled linkability.** When a token is issued, the identifying value y is only known to the customer who generated the token, due to the *partial blindness* of the signature. Hence, if two verified tokens contain the same identifying value y, there are two possibilities: i) both tokens were generated by the same customer, who reused y to allow the vendor to link them; ii) the customer who generated one token leaked y to the customer who generated the other token. If the latter leakage is prevented by technical means or discouraged with appropriate incentives (see discussion in Section 4.6.1), then two tokens containing the same y can be linked by the vendor as corresponding to the same customer.

- **Partial blindness.** In the blinding phase of the partially blind signature of [98], r is randomly chosen from $\mathbb{Z}_q^*$, and thus $u = H_0(c||m)^r$ is a random element of the group $\mathbb{G}_2$. The signer receives this random information and the public information which he already knows. Therefore, no information about the message is leaked.

  The signer is assured that a signature issued by him contains the public information he has agreed on and this information cannot be removed from the signature. This is true because if a malicious user could generate c' and replace c in the signer's signature (c, m, σ) to produce a signature containing c', then the verification equation would be

  $$e(g^{H(c')} \cdot pk, \sigma) \stackrel{?}{=} e(g, H_0(m||c')),$$

  or equivalently

  $$e(g^{(H(c')+x)(H(c)+x)^{-1}}, H_0(c||m)) \stackrel{?}{=} e(g, H_0(c'||m)). \tag{4.1}$$

  Verification (4.1) would only pass for $c' \neq c$ if $H(c') = H(c)$ and $H_0(c'||m) = H_0(c||m)$. This is unlikely, because H, H_0 are cryptographic hash functions.

Finally, due to the randomness introduced during the blinding phase and the fact that the public information is independent of the message, even if the same embedded information is used for two messages, the signer cannot link a signature to the corresponding instance of the signature issuing protocol.

Hence the partially blind signature scheme of [98] satisfies partial blindness.

4.8 Experimental results

We first report execution times of the various protocols from a prototype implementation. We then analyze the deployability of our protocol in stores, by means of two case studies: a physical store and an online store.

4.8.1 Execution times from a prototype

We created a testbed to test the performance of the new protocol in a real environment. Specifically, we wrote a prototype Android customer's application that requests and submits tokens and a vendor application running on a laptop that issues and verifies tokens. To conform to current loyalty program implementations, the communication between customer and vendor is done via NFC using Android's host-card emulation (HCE). Generalization of purchase receipts is done using the Walmart Open API [94] which, on input a product name, returns a list of categories to which the product belongs. The cryptographic protocol was implemented using the jPBC library version 2.0 [36], which runs both in standard and Android Java without further dependencies.

The testbed configuration and parameters are the following. The laptop running the vendor application is an Asus S56C with Intel core i7 3517U, 8GB RAM DDR3 1600Mhz and Ubuntu 15.04. The customer's application is written in Java7 (opendjk-1.7). The NFC reader is an ACS ACR122. The smartphone running the client application is an LG-D821 "Nexus 5" with Android version 5.1, using the ART virtual machine. We generated an asymmetric pairing of Type III, elements in $\mathbb{G}_1$ are 160 bits long, elements in $\mathbb{G}_2$ are 512 bits long.

The first step in a purchase receipt issuance is to obtain the generalization path of the product, from the product name up to the root of the product taxonomy. Using Walmart Open API requests, the mean response time was 1,534 ms. This step does not affect the issuance of loyalty points, and can be drastically reduced if caching mechanisms are used or if the product taxonomy

is stored locally. To test the impact of caching, we implemented a naive caching mechanism, by which responses from the Walmart Open API are stored in a hash table together with the corresponding query. When the vendor makes a query, we check if it was made previously; if yes, we obtain the response from the hash table instead of sending the query to the online service. The tests have shown that the time to obtain the generalization path of a previously queried product name when using caching is reduced to 229 nanoseconds for a universe of 6,167 different products. Clearly, that is a more than affordable time.

Figure 4.3 shows the execution time for the Issuance protocol, broken down by the different stages. The measured total time is 659.1 ms, divided in three stages: blinding, executed by the client application on the customer's smartphone; signing, executed by the vendor on the laptop; and unblinding, again executed by the application on the smartphone. The three stages take, respectively, 356.4 ms, 19.5 ms and 141.7 ms. The difference between the total measured time and the sum of the three stages is 141.5 ms, which corresponds to communication overheads.

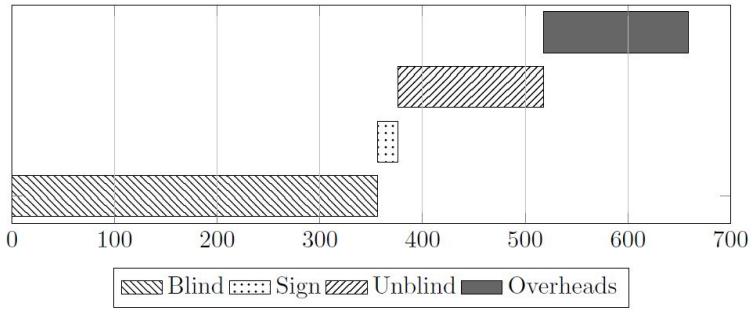

Figure 4.3: Execution times for the Issuance protocol

Figure 4.4 shows the execution time for the Verification protocol. The measured execution time to verify the validity of a token by the vendor is 275.7 ms, while the remaining 401.7 ms is the communication overhead due to the transmission of the whole token (which does not fit in a single APDU).

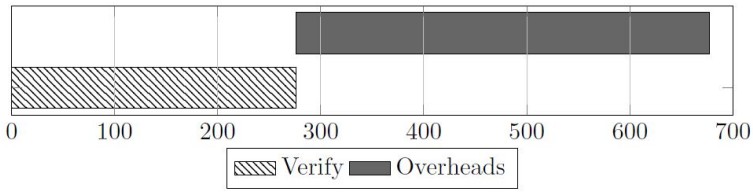

Figure 4.4: Execution times for the Verification protocol

4.8.2 Deployability analysis in stores

We evaluate the applicability of the proposed scheme by means of two case studies: the first one is a physical store and the second is an online store.

Physical store

We model the impact of the application of our solution in physical stores by using queueing theory. By comparing the typical perfomance of the queues at a store with the performance when our solution is applied, we can estimate the additional resources (number of queues) that the store would need to cope with the additional workload incurred by our solution.

Let us set the scenario: each cashier at the physical store is equipped with an NFC reader within whose range the client puts her smartphone as the cashier checks every product. Thus, checking each product and issuing receipt tokens are performed in parallel.

To evaluate the scenario, we need to parameterize it in a realistic way. On the one side, we need to assume a value for the rate λ of customer arrivals per second at the store. To choose a realistic λ, we use Tesco's UK figures, as reported in [2]. In 2014 Tesco lost in the UK 1 million customer visits a week, which decreased their sales by £25m a week; this amounts to £1,300 a year, and this is said to represent 3.1% decrease. Hence, we can deduce Tesco UK receives 32.24 million customers a week, which means 4.6 million customers a day. Now, according to their web, Tesco UK have around 2,500 stores, which results in an average 1,840 customers per store per day; since their opening hours are from 6:00 to 23:00, they receive $\lambda = 0.03006$ customers per store per second. Hence, for any store, the expected inter-arrival time (time between the arrivals of two successive customers) can be estimated as $1/\lambda = 33.7$ seconds.

The actual inter-arrival time can be modeled as an exponential random variable with parameter λ.

The next parameter is the service time, that is, how long it takes the cashier to serve a customer on average. Like the inter-arrival time, the service time can also be viewed as following an exponential distribution, in this case with parameter μ, where $1/\mu$ is the expected service time. We set this time as the number n of products in the customer's basket times the time to check each product (which we estimate at 2 seconds), plus a constant time $c = 30$ s, which is the time devoted to payment. When our protocol is applied, we have to take into account the depth of the generalization tree d and the time to issue a token, which is 0.65 seconds (according to our prototype described in Section 4.8.1). Since token issuance takes place in parallel to product checking, the expected service time is $1/\mu = \max(2, 0.65d)n + 30$. Finally, we assume the number of cashiers in a shop is $q = 4$ (which is a rather modest estimate for large stores such as Tesco's).

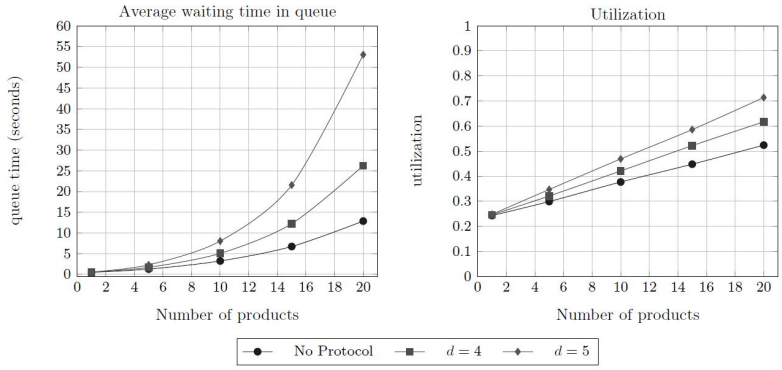

Figure 4.5: Mean waiting time and utilization of cashiers, depending on the number of products per customer and the generalization depth d

In Figure 4.5, we show the mean waiting time at each of the queues/cashiers and the utilization of each of the cashiers as a function of the average number of items in the customers' baskets and the depth of the generalization tree. The blue curve corresponds to three situations: not using the protocol (just bare product checking without token issuance) and also using the protocol with $d = 2, 3$. We label it 'No Protocol' because, even if the protocol is used with

depths 2 or 3, the 2 seconds needed to check each product dominate the time to issue d tokens. We see that the utilization reaches 70% with 20 products and $d = 5$. Although this is acceptable, adding an additional queue might be advisable for those parameter values.

Analytically, assuming an M/M/q model, for token issuance time τ, average number n of products per customer, depth d of the generalization tree, expected inter-arrival time $1/\lambda$ and number of cashiers q, the utilization of each cashier is

$$\rho = \frac{\lambda}{\mu q} = \frac{\lambda(\tau n d + c)}{q}, \tag{4.2}$$

and the mean waiting time is

$$T_w = \frac{C(q, \lambda/\mu)}{\mu q - \lambda} = \left(\frac{\tau n d + c}{q}\right)\left(\frac{C(q, \lambda(\tau n d + c))}{1 - \rho}\right), \tag{4.3}$$

where $C(q, \lambda/\mu)$ is Erlang's C formula.

Results in [89] show that the mean waiting times in a scenario with parallel queues and jockeying (switching queues), which is the typical scenario at a store, are similar (but not equal) to an M/M/q scenario. Formula (4.3), then, gives no more than an approximation to the actual mean waiting time.

Expressions (4.2) and (4.3) are helpful to design a store. For example, given values of τ, n, c and λ for a certain store, and given maximum acceptable values of ρ and T_w (imagine the maximum for ρ is specified by unions and the maximum for T_w by a consumer survey), the store designer can determine the necessary number of cashiers q if using a generalization hierarchy with depth d is desired.

Online store scenario

In an online store transaction, there are typically two main phases: add-to-cart, in which the desired products are added to a list, and checkout, in which the products in the list are actually paid for and purchased.

Our implementation proposal is as follows. Customers adding products to the cart immediately compute the blinding phase of the receipt token issuance for the selected product and all its generalizations. For tree depth d, this step takes $356.4d$ ms of computation by the customer (time according to our prototype described in Section 4.8.1). The online store adds the product to the cart database, as well as the cryptographic material sent by the user. Optionally, the store could execute the signing phase of the token issuing protocol at this

moment, which takes $19.5d$ ms (again according to our prototype). While signing tokens in parallel to product checking has the advantage of making checkout lighter, it might be a waste of resources to sign a receipt token of a product the customer has not yet paid for (because the customer might decide to withdraw one or several products from her basket before checkout). When the customer finally checks out, the store computes (if it has not done so previously) the signature on all receipt tokens and sends them to the customer. At this point, the transaction is already over. The customer can then unblind the signatures on her tokens offline. The additional storage capacity needed by the store is d times the storage required by a token signature per product entry in the cart database ($512d$ bits in our prototype).

In this scenario, we can use metrics for server utilization and average waiting time analogous to the ones we used for physical stores. The main difference is that the number of queues (or servers) q is much more flexible in the online scenario than in the physical scenario (in which each queue required a physical cashier).

Both in a physical and in an online store, the issuance of loyalty points tokens can be analyzed in a way similar to the issuance of receipt tokens, except that loyalty points do not depend on the d factor (whereas d receipt tokens are always issued per product, loyalty points tokens can be issued one at a time). Otherwise, the computation and the storage required to issue one loyalty points token are the same as for one receipt token. Spending loyalty points would be done in the checkout phase, and this would increase the computation time by 275.7 ms per token (the verification time by the store, according to our prototype described in Section 4.8.1).

4.9 Extension for the untransferability of tokens

As mentioned in Sections 4.3.1 and 4.6.1 above, losing loyalty points and purchase privacy is often enough to deter a customer from giving her receipts to another customer. However, in some cases additional deterrence by $\mathcal{V}$ may be needed to prevent purchase receipt transfer and even loyalty points transfer among customers. In this section we propose an extension of our token issuance and verification protocols to enforce untransferability of tokens.

Note that the vendor may be interested in preventing transfer only for one type of tokens. Untransferability is more critical for purchase receipts (because their transfer prevents customer profiling), whereas transferring loyalty points among customers may be tolerable in many situations. If only one type of tokens

needs to be made untransferable, then the extended issuance and verification protocols described in this section should only be applied to that type of tokens.

The intuition is to require the customer to commit to a certain value (*e.g.* the value y) during the token issuance protocol and to require the customer to prove possession of the commitment key during the token verification protocol.

The commitment keys should be securely stored in the device to prevent users from transferring them. If this is achieved, then the tokens are also untransferable. To do so, one might use a trusted platform module installed in the device. For example, Google's Nexus series (up to Nexus 4) carry a TPM integrated in the NFC chip. A more practical solution, and one which is currently gaining popularity among banking applications (and credit card emulation apps) is to use privileged instructions of the device's main processor, a standard ARM extension called TrustZone [90]. Making use of this feature, Android provides a hardware-backed secure keystore API which allows applications to store certificates and keys in a secure way. A key stored by some application can only be retrieved by the same application, even if the device has been compromised. This approach has been used for secure applications such as secure PIN entry, digital rights management, e-ticketing, etc. KNOX [4] is a technology from Samsung, which also uses the TrustZone extension to provide a secure execution environment in mobile devices.

We next specify the changes to the original construction that are needed to guarantee token untransferability:

- Both the setup and the key generation phases remain mostly as described in Sections 4.5.2 and 4.5.2 above, respectively. However, during key generation an additional *commitment key* $sk_{\mathcal{C}} \in_R \mathbb{Z}_q^*$ is generated and given to the customer $\mathcal{C}$ in a secure device (*e.g.* a smart card) such that the key cannot be modified or extracted from the device.

- More changes are needed in the Issuance and Verification protocols with respect to Section 4.5.2. We describe these changes in the following subsections.

4.9.1 Extended Issuance protocol

During this phase, the customer $\mathcal{C}$ commits to the public and the secret information contained in the token with her commitment key, as in Schnorr's ZKP protocol [85]. Namely, Step (3) in Section 4.5.2 is replaced by the following one:

(3*) The customer chooses random $\alpha, r \in_R \mathbb{Z}_q^*$ and builds the message $m = (\alpha, y, h^{sk_C})$, being $h = H_0(c||\alpha||y)$. Thus, h^{sk_C} is a secret commitment on the token information (as the vendor does not know the value α). Then, the customer blinds the message by computing $u = H_0(c||m)^r$ and sends u to the vendor.

The rest of the protocol of Section 4.5.2 remains unaltered. The resulting extended protocol including this modification is shown in Figure 4.6.

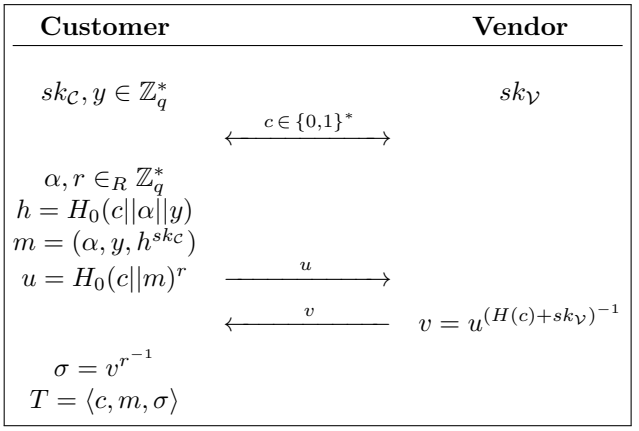

Figure 4.6: Issuance protocol with untransferability

4.9.2 Extended Verification protocol

During this phase, the vendor checks whether the signature on token T was correctly computed (normally by the vendor himself at a previous time) and whether the token has not yet been spent. Additionally, this version of the protocol includes Schnorr's 3-step interactive ZKP [85] to prove knowledge of the commitment key sk_C.

First, and to account for the changes in the contents of the token, replace Step 2 of the Protocol in Section 4.5.2 by the following step:

(2*) The vendor parses the message m as (α, y, h^{sk_u}) and computes $h' = H_0(c||\alpha||y)$.

Then, if the signature verification at Step 3 is correct and the token has not yet been spent, notify the customer to start the interactive ZKP to prove

knowledge of the commitment key. This implies adding the following steps between Steps 3 and 4 of the original protocol

(3a) The customer chooses $k \in_R \mathbb{Z}_q^*$ and sends $t = h^k$ to the vendor.

(3b) The vendor answers with a challenge $sc \in_R \mathbb{Z}_q^*$.

(3c) The customer computes a response $r = k + sk_{\mathcal{C}} \cdot sc$ and sends r to the vendor.

(3d) If $h'^r \stackrel{?}{=} t \cdot h^{sk_{\mathcal{C}} \cdot sc}$ holds, the vendor sends an `accept` message to the customer. Otherwise, the vendor sends a `reject` message.

If the customer can prove knowledge of the commitment key, the vendor is convinced that she participated in the issuance of the token. Hence, the token has not been transferred to a different customer since it was issued. The resulting extended protocol is shown in Figure 4.7.

Customer		**Vendor**
$h, sk_{\mathcal{C}}, T$		$pk_{\mathcal{V}}$
	$\xrightarrow{T=\langle c,m,\sigma\rangle}$	
		parse m as $(\alpha, y, h^{sk_{\mathcal{C}}})$ $h' = H_0(c\|\|\alpha\|\|y)$ $e(g^{H(c)} pk_{\mathcal{V}}, \sigma) \stackrel{?}{=} e(g, H_0(c\|\|m))$
	$\xleftarrow{\text{ok/stop}}$	and token not spent yet
$k \in_R \mathbb{Z}_q^*$ $t = h^k$	$\xrightarrow{t}$	
	$\xleftarrow{sc}$	$sc \in_R \mathbb{Z}_q^*$
$r = k + sk_{\mathcal{C}} \cdot sc$	$\xrightarrow{r}$	
	$\xleftarrow{\text{accept/reject}}$	$h'^r \stackrel{?}{=} t \cdot h^{sk_{\mathcal{C}} \cdot sc}$
		Link token to previous ones with same y

Figure 4.7: Verification protocol with untransferability

4.9.3 Complexity and security analysis of the extension

Regarding complexity, we evaluate the new computations added by the extension. In the new Step (3*) of the Issuance protocol a commitment is now computed. This requires the customer to compute one additional hash and one additional exponentiation in $\mathbb{Z}_q^*$.

In the new steps added to the Verification protocol, the following additional computations are needed:

- The customer computes a hash in Step (2*), a random number and an exponentiation in $\mathbb{Z}_q^*$ in Step (3a) and a multiplication in Step (3c).
- The vendor computes a random challenge in Step (3b) and two exponentiations and a multiplication in $\mathbb{Z}_q^*$ in Step (3d).

If one compares these new computations with the computations in the original protocols before the extension, it can be inferred that the computing time does not even double as a result of the extension, neither for the customer nor the vendor. Hence, it remains affordable.

The security of the extension is based on:

- The security of Schnorr's ZKP, proven in [85, 70]. Hence, a customer not holding the commitment key cannot convince a vendor that she holds the key. Also, neither the vendor nor any observer learn any bit of the commitment key.
- The tamper-resistance of the secure device. This device performs all customer computations involving the commitment key, in such a way that the customer cannot learn the value of her own key. This ensures that no customer can transfer her commitment key to another customer. Hence, when a customer proves during token verification that she holds the same commitment key used for token issuance, the vendor is convinced that the token has not been transferred.

4.10 Summary

In our privacy-aware alternative to traditional loyalty programs, the customers are granted the power to decide what private information they want to disclose, and how accurate that information is. We have described a privacy-aware protocol suite that still offers the two main features of loyalty programs: to

reward returning customers and to make customer profiling possible. We have presented experimental results that show that our suite is usable in practice. Finally, we have proposed an extension of our protocol suite which prevents purchase receipts and loyalty points from being transferred among customers; if only transfer of one type of tokens is to be prevented, then our extension need only be used to issue and verify that type of tokens.

Chapter 5

Privacy-preserving implicit authentication

5.1 Introduction

Implicit authentication refers to a software system authenticating individuals based on the way they interact with their device, *i*.e. their behavior. In this context, the user behavior can be determined by collecting a variety of features, such as keystroke patterns, browser history and configuration, IP addresses, location, visible antennas, etc. Implicit authentication can be viewed as a complement of the usual explicit authentication based on identifiers and credentials.

Implicit authentication is gaining importance as the smartphone market rises. Relatively small and sometimes unwieldy screen keyboards in smartphones make typing strong passwords a difficult task. This situation, added to the well-known problem of weak password choices, repeatedly aired in the media, makes the use of secondary authentication mechanisms almost mandatory. Among these, biometric (fingerprint) authentication and two-factor authentication with one-time passwords are the most common choices. Biometric authentication has the shortcomings of needing special sensors in the user's device and requiring the authenticating server to acquire and store the user's reference biometric pattern. Two-factor authentication, on its side, has an intrinsic problem: the second channel (email, SMS, mobile app) used for confirmation is usually accessible on the same device (typically an Internet-enabled smartphone) used for the primary channel, so both channels may be simultaneously compromised.

Implicit authentication is not free of problems either. A salient issue is the privacy exposure of end users, who need to be profiled in order to provide a reference pattern against which their current behavior can be authenticated by the server.

5.1.1 Contributions

We propose in here two privacy-preserving implicit authentication protocols based on the computation of distances between feature sets. We justify that these distances can be obtained from the size of the intersection of the feature sets and use two different mechanisms from the literature on record matching schemes to compute the intersections in a private way. Thus, our first mechanism is based on the protocol from [49], of which we provided an overview in Section 2.6. The second one is computationally lighter, since it is based on the intersection of Bloom filters (described in Section 2.8).

The contributions in this chapter have been published in [46, 15, 43]. The implicit authentication construction based on Bloom filters has been submitted to a journal.

This Chapter is organized as follows. First, in Section 5.2 we give an overview of the past related works in implicit authentication and privacy-preserving implicit authentication. In Section 5.3 we state the implicit authentication problem, including a description of the scenario, user profiles and possible threats. Then, in Section 5.4 we describe the types of feature sets that model the user preferences or behavior, and that are supported by our distance computation mechanism and our implicit authentication mechanism. In Section 5.6 we propose a robust privacy-preserving implicit authentication mechanism based on the homomorphic properties of the Paillier cryptosystem. Sections 5.7 and 5.8 justify the security and privacy guarantees of the protocol and provide experimental results, respectively. Section 5.9 describes an alternative privacy-preserving implicit authentication mechanism, based, in this case, on Bloom filters and their properties. In Section 5.10 and 5.11 we discuss the security and privacy of the protocol, and provide experimental results, respectively. Finally, Section 5.12 presents the final remarks on our two proposed privacy-preserving implicit authentication protocols.

5.2 Related work on implicit authentication

We first specify the usual setting of implicit authentication and we then move to privacy-preserving implicit authentication.

5.2.1 Implicit authentication

The usual scenario of implicit authentication is one in which the user carries a mobile networked device (called just user's device in what follows) such as a cell phone, tablet, notebook, etc. The user wishes to authenticate to a server in order to use some application. The user may (or not) use a primary password authentication mechanism. To strengthen such a primary authentication or even to replace it, the user resorts to *implicit authentication* [58]. In this type of authentication, the history of a user's actions on the user's device is used to construct a profile for the user that consists of a set of features. In [58] empirical evidence was given that the features collected from the user's device history are effective to distinguish users and therefore can be used to implicitly authenticate them (instead or in addition to explicit authentication based on the user's providing a password).

The types of features collected on the user's actions fall into three categories: (i) device data, like GPS location data, WiFi/Bluetooth connections and other sensor data; (ii) carrier data, such as information on cell towers seen by the device, or Internet access points; and (iii) cloud data, such as calendar entries. It is not safe to store the accumulated profile of the user in the user's device, because an intruder might compromise the device and alter the stored profile in order to impersonate the legitimate user. Hence, for security, the profile must be stored by some external entity. However, the user's profile includes potentially sensitive information and storing it outside the user's device violates privacy.

Implicit authentication systems try to mitigate the above privacy problem by using a third party, the *carrier* (i.e. the network service provider) to store the user's profiles. Thus, the typical architecture consists of the user's device, the carrier and the application servers. The latter want to authenticate the user and they collaborate with the carrier and the user's device to do so. The user's device engages in a secure two-party computation protocol with the carrier in order to compare the fresh usage features collected by the user's device with the user's profile stored at the carrier. The computation yields a score that is compared (by the carrier or by the application server) against a threshold, in order to decide whether the user is accepted or rejected. In any case, the application server trusts the score computed by the carrier.

5.2.2 Privacy-preserving implicit authentication

In the privacy-preserving implicit authentication system proposed in [82], the user's device encrypts the user's usage profile at set-up time, and forwards it to the carrier, who stores it for later comparison. There is no security problem because during normal operation the user's device does not store the user's profile (it just collects the fresh usage features). There is no privacy problem either, because the carrier does not see the user's profile in the clear.

The core of proposal [82] is the algorithm for computing the dissimilarity score between two inputs: the fresh sample provided by the user's device and the profile stored at the carrier. All the computation takes place at the carrier and both inputs are encrypted: indeed, the carrier stores the encrypted profile and the user's device sends the *encrypted* fresh sample to the carrier. Note that the keys to both encryptions are only known to the user's device (it is the device who encrypted everything).

The carrier computes a dissimilarity score at the feature level, while provably guaranteeing that: i) no information about the profile stored at the carrier is revealed to the device other than the average absolute deviation of the stored feature values; ii) no information about the fresh feature value provided by the device is revealed to the carrier other than how it is ordered with respect to the stored profile feature values.

The score computation protocol in [82] uses two different encryption schemes: a homomorphic encryption scheme HE (for example, Paillier's [74]) and an order-preserving symmetric encryption scheme $OPSE$ (for example, [20]). For each feature in the accumulated user's profile, two encrypted versions are created, one under HE and the other under $OPSE$. Similarly, for each feature in the fresh sample it collects, the user's device computes two encrypted versions, under HE and $OPSE$, respectively, and sends them to the carrier. The following process is repeated for each feature:

1. Using the HE ciphertexts the carrier performs some computations (additions and scalar multiplications) relating the encrypted fresh sampled feature value and the set of encrypted feature values in the stored encrypted user's profile.

2. The output of the previous computations is returned to the user's device, which decrypts it, re-encrypts it under $OPSE$ and returns the re-encrypted value to the carrier.

3. Using the order-preserving properties, the carrier can finally compute a dissimilarity score evaluating how different is the fresh sampled feature

from those stored in the encrypted user's profile. This score can be roughly described as the number of feature values in the stored encrypted profile that are less dissimilar from the median of the stored values than the fresh sampled value.

The authors of [82] point out that, in case of a malicious user's device (*e.g.* as a result of it being compromised), one cannot trust the device to provide the correct HE-encrypted version of the fresh sampled feature. Nor can it be assumed that the device returns correct $OPSE$-encryptions in Step 2 above. In [82], a variant of the privacy-preserving implicit authentication protocol is presented in which the device proves the correctness of HE-encrypted fresh sampled features and does not need to provide $OPSE$-encrypted values. This version is secure against malicious devices, but its complexity is substantially higher.

Other shortcomings of [82]:

- It is restricted to numerical features, due to the kind of computations that need to be performed on them. However, among the example features listed in Section 5.2.1, there are some features that are not numerical, like the list of cell towers or Internet access points seen by the user's device.

- It discloses the following information to the user's device: i) how the fresh sample is ordered with respect to the stored profile feature values; ii) the average absolute deviation of the stored feature values.

5.3 Preliminary definitions

5.3.1 Scenario

We consider a scenario in which smartphone users log into online services offered by some service provider. Users set their login information at registration time, and this information is managed by either the service provider or by a third party (an identity provider that delivers authentication services). The entity in charge of authenticating users offers an additional security measure: it analyzes the behavior of users to detect potentially compromised user accounts. The service provider denies service to accounts labeled as compromised, and the rightful owners of those accounts are notified.

5.3.2 User profiles, behavior samples and feature sets

In this section, we describe the user profiles as we will use them throughout the rest of this work. A **user profile** is a list of snapshots of the user's behavior at different times. These snapshots, or **samples**, are labeled by a timestamp and contain one or more feature sets. Each **feature set** contains readings from one specific data source in the user's device for a fixed period of time.

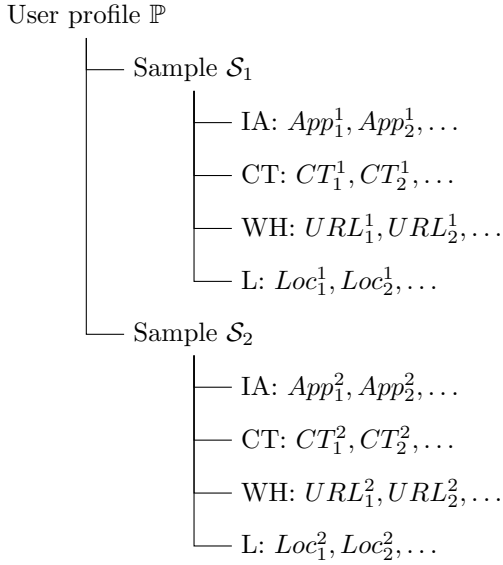

Figure 5.1: User profile with two behavior samples. Within each sample, feature sets are as follows: installed applications (IA), visible cell towers (CT), web browsing history (WH), visited locations (L).

Figure 5.1 shows an example of a user profile $\mathbb{P}$ with two samples $\mathcal{S}_1$ and $\mathcal{S}_2$, each of them containing several feature sets S_i (each of them labeled according to the data it contains), namely installed applications, visible cell towers, web browsing history and visited locations. The label $\mathcal{S}_t$ refers to specific sample and the subscript t may refer to the time of collection (a timestamp). Each of the feature sets includes information gathered from different sources during some predefined period of time; for example, the set labeled as CT in sample $\mathcal{S}_1$ may be the set of cell towers that the device has seen during the last day.

Typical data sources considered for authentication are, among others, installed applications, installed applications by category, usage of applications, usage of applications by category, visible cell towers, strength of the signal of cell towers, battery level at the time the device is connected for charging, time between consecutive charges, power consumption, idle and awake times, web browsing history, web browsing history by category, trajectories of the user, etc. Some of these features may be considered more important than others when taking an authentication decision; hence, different weights may be assigned to the various features.

In the implicit authentication protocol, the user sends a new sample to the server, who compares it to ℓ previous samples, feature set by feature set, to reach an authentication decision. It is also worth considering that, since users may behave differently in work days and weekends, samples may need to be compared with parts of the profile corresponding to similar days and times. New samples that result in successful authentications can be stored by the server to update the user profile.

Assuming that every feature has the same weight when taking the authentication decision (*e.g.*, installed applications matter as much as visited locations), our construction allows representing each sample as a single feature set of the following form (where each value is labeled with the name of the feature set it comes from):

- $\mathcal{S}_1$: $\{\text{IA:}App_1^1, \text{IA:}App_2^1, \ldots, \text{CT:}CT_1^1, \ldots, \text{WH:}URL_1^1, \ldots, \text{L:}Loc_1^1, \ldots\}$
- $\mathcal{S}_2$: $\{\text{IA:}App_1^2, \text{IA:}App_2^2, \ldots, \text{CT:}CT_1^2, \ldots, \text{WH:}URL_1^2, \ldots, \text{L:}Loc_1^2, \ldots\}$

In this case, the implicit authentication protocol can run a single set comparison per sample. However, if the weights of the features differ, or the profiles contain categorical *and* numerical feature sets, the authentication protocol will have to compute the distances between several sets. An example of this situation is shown in Figure 5.2.

In this case, to compare the sample S_3 with older samples (of the same form) in an authentication attempt, the following sets have to be compared:

- $\{\text{IA:}App_1^3, \text{IA:}App_2^3, \ldots, \text{CT:}CT_1^3, \text{CT:}CT_2^3, \ldots\}$
- $\{\text{WH:}URL_1^3, \text{WH:}URL_2^3, \ldots, \text{L:}Loc_1^3, \text{L:}Loc_2^3, \ldots\}$
- $\{K_1^3, K_2^3, \ldots\}$

While the mechanisms for the collection of data and the data sources themselves are outside of the scope of this work, we impose the requirement that

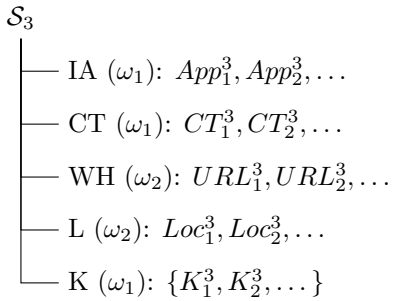

Figure 5.2: Behavior sample with different weights (denoted by ω_1 and ω_2). Categorical feature sets: installed applications (IA), visible cell towers (CT), web browsing history (WH), visited locations (L). Numerical feature set: kilometers walked each hour (K).

features must be discretized, for example via generalization/coarsening. Take location data as an example: GPS location data are too fine-grained for comparison using our distance computation mechanism, so we require them to be coarsened either by translating the latitude-longitude pairs to names (*e.g.* street names), or by truncating some decimals (which in practice amounts to defining a grid in a map).

5.3.3 Privacy attacker

We regard any entity with legitimate or illegitimate access to the stored user profiles as a potential privacy attacker. An attacker seeing the profile of a user learns sensitive information about the user, such as her typical locations, her preferences, the software installed on her computer, etc. The user's profile may also allow inferring the user's identity, in case the latter is not readily available. Hence, user profiles must be protected while in transit and when stored on the server premises.

5.3.4 Impersonator

An attacker who gains access to a user's device, her account credentials (*e.g.* login and password) or both, may try to impersonate the user by accessing her accounts. By analyzing the behavior of the impersonator it should be possible to detect the attack and deny access to the impersonator.

5.4 Dissimilarity between feature sets depending on data types

We present in here how the dissimilarity between two feature sets X and Y can be evaluated using set intersection. If we let X be the user's profile and Y be the fresh sample collected by the user's device, our privacy-preserving implicit authentication mechanisms aim at computing the distances of these sets in some protected form. We describe here the case of two plaintext sets X and Y and we will deal with protected sets in the following sections.

5.4.1 Case A: independent nominal feature values

Consider two feature sets X and Y containing independent categorical values, such that the relationship between any two values is equality or nothing. These sets might for example represent the user's browser history (containing only domain names, but not specific pages, because specific pages would be clearly correlated to their domains), visible cell towers, installed applications, etc. The (dis)similarity between X and Y can be computed as the multiplicative inverse of the size of their intersection, that is $1/|X \cap Y|$, or ∞ when the intersection is empty. Note that Bloom filters allow computing also the cardinality of the union of two sets, which makes it possible, in our case, to compute the Jaccard similarity index $J(X,Y) = |X \cap Y|/|X \cup Y|$ or its complement, the Jaccard distance, that is $d_J(X,Y) = 1 - J(X,Y) = (|X \cup Y| - |X \cap Y|)/|X \cup Y|$. This latter distance, being normalized, is a very convenient measure.

Clearly, the more the coincidences between X and Y, the more similar is the profile stored at the server to the fresh sample collected by the device.

5.4.2 Case B: correlated categorical feature values

As in the previous case, we assume the feature values are expressed as qualitative features. However, these may not be independent. For example, if the feature values are the IDs of cell towers or Internet access points seen by the device, nearby cell towers/access points are more similar to each other than distant cell towers/access points.

In this case, the dissimilarity between X and Y cannot be computed as the size of their intersection.

Assume we have an integer correlation function $l : E \times E \mapsto \mathbb{Z}_+$ that measures the similarity between the values in the sets of features held by the device and the carrier, where E is the domain where the sets of features of both players take

values. For nominal features, semantic similarity measures can be used for this purpose [83]; for numerical features that take values over bounded and discrete domains, standard arithmetic functions can be used. Assume further that both the device and the carrier know this function s from the very beginning.

Here the dissimilarity between the set X and the set Y can be computed as

$$1/(\sum_{x \in X} \sum_{y \in Y} l(x, y))$$

when the denominator is nonzero. If it is zero, we say that the distance is ∞.

5.4.3 Case C: numerical feature values

In this case, the profile of the user is a set of numerical values, for example, sensor data, the browser history expressed as the number of accesses to each website in a list, a histogram of user preferences, etc.

Given two sets $X = \{x_1, \cdots, x_t\}$ and $Y = \{y_1, \cdots, y_t\}$, a way to measure the dissimilarity between them is to compute $\sum_{i=1}^{t} |x_i - y_i|$. If X and Y represent normalized histograms, that is, $0 \leq x_i \leq 1$ for all i and $\sum_{i=1}^{t} x_i = 1$ (or 100 if the values are given as percentages), we could also normalize the resulting distance, because the maximum possible distance is known (2 if the features of each histogram add to 1).

5.5 Proposed architecture

The high-level architecture of both our implicit authentication mechanisms is centered on a protection module in the user's smartphone that collects sensor data and protects the samples, and a distance computation module on the server's side that can compute the (dis)similarity of the protected fresh sample and the protected recorded user profile, and return an authentication score or decision. We propose two alternative architectures built on these core modules. The first one, shown in Figure 5.3 is a traditional approach, in which the service provider is in charge of authenticating its users; the second one, shown in Figure 5.4 follows a single sign-on approach, in which an identity provider, for example the carrier, is in charge of authenticating the users on behalf of the service provider.

Next, we briefly describe the two above-mentioned core modules. Then, the interaction between the protection module and the distance computation module, that is, the implicit authentication protocol, is described in Sections 5.6 and 5.9, which account for our two proposed mechanisms.

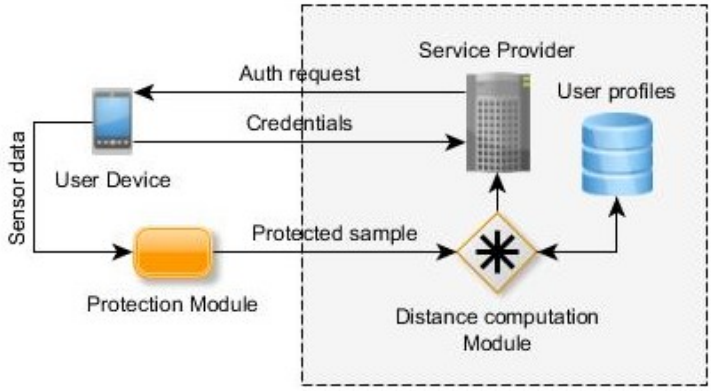

Figure 5.3: Basic architecture

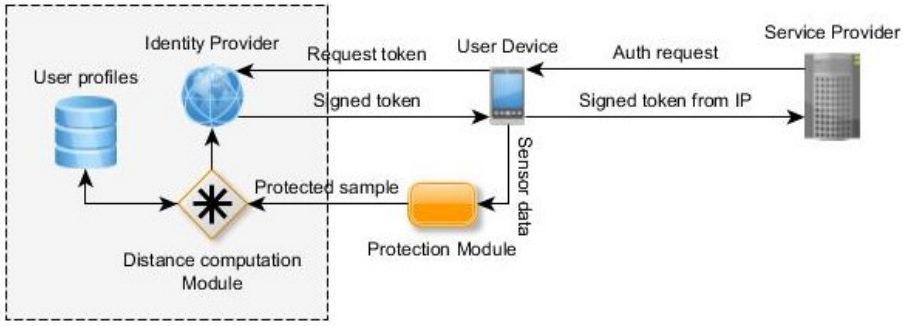

Figure 5.4: Architecture with an identity provider (IP)

Protection module

The protection module in the user's device is a software module that first gathers data from the device. These data, as pointed out in previous sections, are sensor data, network statistics, installed applications, browser histories, etc. They are used to build the fresh sample of the user profile that will be the input to implicit authentication. Then the protection module protects the sampled profile by encoding it into a Bloom filter. Finally, the module sends the Bloom filter to the server.

Distance computation module

The distance computation module runs on the server side and is capable of comparing protected sampled profiles against previously stored protected reference user profiles. The comparison returns an authentication score that is compared to a threshold, in order to output an authentication decision.

5.6 Privacy-preserving implicit authentication from Paillier

In this section we show that computing dissimilarities in the above three cases A, B and C can be reduced to computing the cardinality of set intersections. Furthermore, this can be done without the carrier revealing X and without the user's device revealing Y, as required in the implicit authentication setting. The idea is that, if the dissimilarity stays below a certain threshold, the user is authenticated; otherwise, authentication is refused.

In Chapter 2, we gave some background on privacy-preserving set intersection protocols in the literature. Unfortunately, most of them assume an honest-but-curious situation, but we need a privacy-preserving set intersection protocol that works even if the adversary is a malicious one: notice that the user's device may be corrupted, that is, in control of some adversary. Hence we proceed to specifying a set intersection protocol that remains robust in the malicious scenario and we apply it to achieving privacy-preserving implicit authentication in Case A. We then extend it to Cases B and C. We make use of Paillier's cryptosystem [74], which is recalled in Chapter 2.

5.6.1 Implicit authentication in case A

Set-up

Let the plaintext user's profile be $(a_1, \cdots, a_s)$. In this phase, the user's device transfers the encrypted user's profile to the carrier. To do so, the user's device does:

1. Generate the Paillier cryptosystem with public key $pk = (n, g)$ and secret key sk.
2. Compute the polynomial $p(x) = \prod_{i=1}^{s}(x - a_i) = p_0 + p_1 x + p_2 x^2 + \cdots + p_s x^s$.
3. Compute $Enc(p_0), \cdots Enc(p_s)$ where $Enc(p_i) = g^{p_i} r_i^n \mod n^2$.
4. Randomly choose $R' \in Z_{n^2}$. Find $r'_0, \cdots, r'_s \in Z_{n^2}$ such that

$$R' = r'_0 \cdot {r'_1}^{a_j} \cdot {r'_2}^{a_j^2} \cdots {r'_s}^{a_j^s} \mod n^2, \quad j = 1, \cdots, s. \tag{5.1}$$

   Note that the system (5.1) has a trivial solution $r'_0 = R'$ and $r'_1 = \cdots = r'_s = 1$, but, since it is underdetermined ($s + 1$ unknowns and s equations), it has many non-trivial solutions too (see correctness analysis in Section 5.7).

5. Compute $R_i = r'_i / r_i \mod n^2$. Randomly choose integer $d \in Z_n$. Send

$$pk, Enc(p_0), \cdots Enc(p_s); {R_0}^d, \cdots, {R_s}^d \mod n^2$$

   to the carrier. Locally delete all data computed during the set-up protocol, but keep (d, R') secretly.

Implicit authentication protocol

As discussed in Section 5.4.1, in case of independent nominal feature values (Case A), dissimilarity is computed as $1/|X \cap Y|$. Hence, to perform implicit authentication the carrier just needs to compute the cardinality of the intersection between the fresh sample collected by the user's device and the user's profile stored at the carrier. *The challenge is that the carrier only holds the encrypted user's profile and the user's device does no longer hold the plaintext user's profile either in plaintext or ciphertext.*

Let $Y = \{b_1, \cdots, b_t\} \subseteq E$ be the fresh sample collected by the user's device. Then the device and the carrier engage in the following protocol:

Step 1 The carrier randomly chooses θ, and sends pk, $Enc(p_0)^\theta$, $\cdots$ $Enc(p_s)^\theta$; ${R_0}^d$, $\cdots$, ${R_s}^d$ to the user's device.

Step 2 The user's device picks a random integer $r(j) \in \mathbb{Z}_{n^2}$ for every $1 \leq j \leq t$. The device computes for $1 \leq j \leq t$

$$\begin{aligned} Enc(r(j) \cdot d \cdot \theta \cdot p(b_j)) &= Enc(p(b_j))^{d \cdot \theta \cdot r(j)} \\ &= (Enc(p_0) \cdots Enc(p_s)^{b_j^s})^{d \cdot \theta \cdot r(j)} \\ &= g^{r(j) \cdot d \cdot \theta p(b_j)} \gamma_j^{n \cdot d \cdot \theta} \mod n^2 \end{aligned}$$

where $\gamma_j = (r_0 \cdot {r_1}^{b_j} \cdot {r_2}^{b_j^2} \cdots {r_s}^{b_j^s})^{r(j)} \mod n^2$. The user's device then computes $\Upsilon_j = (R_0 \cdot {R_1}^{b_j} \cdot {R_2}^{b_j^2} \cdots {R_s}^{b_j^s})^{dr(j)} \mod n^2$. For all j, the device randomly orders and sends

$$\{(Enc(r(j) \cdot d \cdot \theta \cdot p(b_j)), \Upsilon_j, {R'}^{r(j)d})\} \tag{5.2}$$

to the carrier.

Step 3 For $1 \leq j \leq t$, the carrier does:

- Compute $Enc(r(j) \cdot d \cdot \theta \cdot p(b_j)) \cdot \Upsilon_j^{n\theta}$;
- From Expression (5.1), if $b_j = a_i$ for some $i \in \{1, \cdots, s\}$, then $p(b_j) = 0$ and hence $Enc(r(j)d \cdot \theta \cdot p(b_j)) \cdot \Upsilon_j^{n\theta} = {R'}^{r(j)dn\theta}$; note that the carrier can recognize ${R'}^{r(j)dn\theta}$ by raising ${R'}^{r(j)d}$ received in Expression (5.2) to $n\theta$. Otherwise (if $b_j \neq a_i$ for all $i \in \{1, \cdots, s\}$) $Enc(r(j) \cdot d \cdot \theta \cdot p(b_j))$ looks random. See correctness analysis in Section 5.7.

If both parties are honest, then the carrier learns $|X \cap Y|$ but obtains no information about the elements in X or Y.

5.6.2 Implicit authentication in case B

Here, the carrier inputs X and the user's device inputs Y, two sets of features, and they want to know how close X and Y are without revealing their own set. In the protocol below, only the carrier learns how close X and Y are.

We assume that the domain of X and Y is the same, and we call it E. The closeness or similarity between elements is computed by means of a function s. In particular, we consider functions $l : E \times E \to \mathbb{Z}_+$. Observe that Case A is a particular instance of this Case B in which $l(x, x) = 1$ and $l(x, y) = 0$ for $x \neq y$.

Let Y be the input of the user's device. For every $z \in E$, the device computes $\ell_z = \sum_{y \in Y} l(z, y)$. Observe that ℓ_z measures the overall similarity of z and Y. Let $Y' = \{z \in E : \ell_z > 0\}$. It is common to consider functions satisfying $l(z, z) > 0$ for every $z \in E$, and so in general $Y \subseteq Y'$.

An implicit authentication protocol for such a computation can be obtained from the protocol in Case A (Section 5.6.1), by replacing Steps 2 and 3 there with the following ones:

Step 2' For every $z \in Y'$, the user's device picks ℓ_z random integers $r(1), \cdots, r(\ell_z) \in \mathbb{Z}_{n^2}$ and for $1 \leq j \leq \ell_z$ does

- Compute

$$\begin{aligned} Enc(r(j) \cdot d \cdot \theta \cdot p(z)) &= Enc(p(z))^{d \cdot \theta \cdot r(j)} \\ &= (Enc(p_0) \cdots Enc(p_s)^{z^s})^{d \cdot \theta \cdot r(j)} \\ &= g^{r(j) \cdot d \cdot \theta p(z)} \gamma_j^{n \cdot d \cdot \theta} \mod n^2 \end{aligned}$$

  where $\gamma_j = (r_0 \cdot {r_1}^z \cdot {r_2}^{z^2} \cdots {r_s}^{z^s})^{r(j)} \mod n^2$.
- Compute $\Upsilon_j = (R_0 \cdot {R_1}^z \cdot {R_2}^{z^2} \cdots {R_s}^{z^s})^{dr(j)} \mod n^2$.
- Let $E_j = \{(Enc(r(j) \cdot d \cdot \theta \cdot p(z)), \Upsilon_j, {R'}^{r(j)d})\}$.

Finally, the user's device randomly re-orders the sequence of all computed E_j for all $z \in Y'$ (a total of $\sum_{z \in Y'} \ell_z$ elements) and sends the randomly re-ordered sequence of E_j's to the carrier.

Step 3' For every received E_j, the carrier does

- Compute $Enc(r(j)d\theta \cdot p(z)) \cdot \Upsilon_j^{n\theta}$;
- From Expression (5.1), if $z \in X$, then $p(z) = 0$ and hence $Enc(r(j)d \cdot \theta \cdot p(z)) \cdot \Upsilon_j^{n\theta} = {R'}^{r(j)dn\theta}$ (see correctness analysis in Section 5.7); otherwise (if $z \notin X$) $Enc(r(j)d\theta \cdot p(z))$ looks random.

Hence, at the end of the protocol, the total number of E_j which yield ${R'}^{r(j)dn\theta}$ is

$$\sum_{x \in X} \ell_x = \sum_{x \in X} \sum_{y \in Y} l(x, y),$$

that is, the sum of similarities between the elements of X and Y. This clearly measures how similar X and Y are. At the end of the protocol, the carrier knows $|Y'|$ and the device knows $|X|$. Besides that, neither the carrier nor the device can gain any additional knowledge on the elements of each other's set of preferences.

5.6.3 Implicit authentication in case C

Let the plaintext user's profile be a set U of t numerical features, which we denote by $U = \{u_1, \cdots, u_t\}$. The device's fresh sample corresponding to those features is $V = \{v_1, \cdots, v_t\}$. The carrier wants to learn how close X and Y are, that is, $\sum_{i=1}^{t} |u_i - v_i|$.

Define $X = \{(i,j) : u_i > 0 \text{ and } 1 \leq j \leq u_i\}$ and $Y = \{(i,j) : v_i > 0 \text{ and } 1 \leq j \leq v_i\}$. Now, take the set-up protocol defined in Section 5.6.1 for Case A and run it by using X as plaintext user profile. Then take the implicit authentication protocol for Case A and run it by using Y as the fresh sample input by the device. In this way, the carrier can compute $|X \cap Y|$. Observe that

$$|X \cap Y| = |\{(i,j) : u_i, v_i > 0 \text{ and } 1 \leq j \leq \min\{u_i, v_i\}\}| = \sum_{1 \leq i \leq t} \min\{u_i, v_i\}.$$

In the set-up protocol for Case A, the carrier learns $|X|$ and during the implicit authentication protocol for Case A, the carrier learns $|Y|$. Hence, the carrier can compute

$$\begin{aligned} |X| + |Y| - 2|X \cap Y| &= \sum_{i=1}^{t} (\max\{u_i, v_i\} + \min\{u_i, v_i\}) - 2\sum_{i=1}^{t} \min\{u_i, v_i\} \\ &= \sum_{i=1}^{t} (\max\{u_i, v_i\} - \min\{u_i, v_i\}) = \sum_{i=1}^{t} |u_i - v_i|. \end{aligned}$$

5.7 Security, privacy and complexity

5.7.1 Correctness

In general, the correctness of our protocol follows from direct algebraic verification using the properties of Paillier's cryptosystem. We go next through the least obvious steps.

Set-up protocol

In the set-up protocol, $r'_0, \cdots, r'_s$ are found as a solution of the following system

$$\begin{bmatrix} R' \\ \vdots \\ R' \end{bmatrix} = \begin{bmatrix} r'_0 \cdot {r'_1}^{a_1} \cdot {r'_2}^{a_1^2} \cdots {r'_s}^{a_1^s} & \mod n^2 \\ \vdots & \\ r'_0 \cdot {r'_1}^{a_s} \cdot {r'_2}^{a_s^2} \cdots {r'_s}^{a_s^s} & \mod n^2 \end{bmatrix}.$$

The above system has $s+1$ unknowns and s equations. Therefore it has one degree of freedom. To avoid the trivial solution $r'_0 = R'$ and $r'_1 = \cdots = r'_s = 1$, we choose a random r'_0. Then we divide the system by r'_0 and we take logarithms to get

$$\begin{bmatrix} \log(R'/r'_0) \\ \log(R'/r'_0) \\ \vdots \\ \log(R'/r'_0) \end{bmatrix} \mod n = \begin{bmatrix} a_1 & a_1^2 & \cdots & a_1^s \\ \vdots & \vdots & \vdots & \\ a_s & a_s^2 & \cdots & a_s^s \end{bmatrix} \cdot \begin{bmatrix} \log r'_1 \\ \log r'_2 \\ \vdots \\ \log r'_s \end{bmatrix} \mod n.$$

The matrix on the right-hand side of the above system is an $s \times s$ generalized Vandermonde matrix (not quite a Vandermonde matrix). Hence, using the techniques in [38] it can be solved in $O(s^2)$ time for $\log r'_1, \cdots, \log r'_s$. Then s powers modulo n^2 need to be computed to turn $\log r'_i$ into r'_i for $i = 0, \cdots, s$.

Implicit authentication protocol

We specify in more detail the following derivation in Step 2 of the implicit authentication protocol of Section 5.6.1:

$$\begin{aligned} Enc(r(j) \cdot d \cdot \theta \cdot p(b_j)) &= Enc(p(b_j))^{d \cdot \theta \cdot r(j)} \mod n^2 \\ &= (Enc(p_0) \cdots Enc(p_s)^{b_j^s})^{d \cdot \theta \cdot r(j)} \mod n^2 \\ &= (g^{p_0} r_0^n \cdots (g^{p_s} r_s^n)^{b_j^s})^{d \cdot \theta \cdot r(j)} \mod n^2 \\ &= (g^{p(b_j)})^{d \cdot \theta \cdot r(j)} (r_0 \cdot r_1^{b_j} \cdots r_s^{b_j^s})^{r(j) \cdot n \cdot d \cdot \theta} \mod n^2 \\ &= g^{r(j) \cdot d \cdot \theta p(b_j)} \gamma_j^{n \cdot d \cdot \theta} \mod n^2. \end{aligned}$$

Regarding Step 3 of the implicit authentication protocol, we detail the case $b_j = a_i$ for some $i \in \{1, \cdots, s\}$. In this case, $p(b_j) = 0$ and hence

$$\begin{aligned} Enc(r(j) d\theta \cdot p(b_j)) \cdot \Upsilon_j^{n\theta} \mod n^2 &= Enc(0)^{r(j) d\theta} \cdot \Upsilon_j^{n\theta} \mod n^2 \\ &= (r_0 \cdot r_1^{b_j} \cdots r_s^{b_j^s})^{nr(j) d\theta} \cdot \Upsilon_j^{n\theta} \mod n^2 \\ &= (r_0 \cdot r_1^{b_j} \cdots r_s^{b_j^s})^{nr(j) d\theta} \cdot (R_0 \cdot R_1^{b_j} \cdots R_s^{b_j^s})^{dr(j) n\theta} \mod n^2 \\ &= (r'_0 \cdot r'^{a_i}_1 \cdots r'^{a_i^s}_s)^{r(j) dn\theta} \mod n^2 = R'^{r(j) dn\theta} \mod n^2. \end{aligned} \quad (5.3)$$

If in Step 3, if we have $b_j \neq a_i$ for all $i \in \{1, \cdots, s\}$, then Derivation (5.3) does not hold and a random number is obtained instead. On the one side, the powers of g does not disappear from $Enc(r(j) d\theta \cdot p(b_j))$. On the other side, the exponents $b_j, \cdots, b_j^s$ cannot be changed by $a_i, \cdots, a_i^s$ as done in the last step of Derivation (5.3). Hence, a random number different from $R'^{r(j) dn\theta}$ is obtained.

5.7.2 Privacy and security

Unless otherwise stated, the assessment in this section will focus on the protocols of Case A (Section 5.6.1), the protocols of Cases B and C being extensions of Case A.

We define privacy in the following two senses:

- After the set-up is concluded, the user's device does not keep any information about the user's profile sent to the carrier. Hence, compromise of the user's device does not result in compromise of the user's profile.
- The carrier learns nothing about the plaintext user's profile, except its size. This allows the user to preserve the privacy of her profile towards the carrier.

Lemma 1. *After set-up, the user's device does not keep any information on the user's profile sent to the carrier.*

Proof. The user's device only keeps (d, R') at the end of the set-up protocol. Both d and R' are random and hence unrelated to the user's profile. □

Lemma 2. *The carrier or any eavesdropper learn nothing about the plaintext user's profile, except its size.*

Proof. After set-up, the carrier receives $pk, Enc(p_0), \cdots Enc(p_s); {R_0}^d, \cdots, {R_s}^d \mod n^2$. Since d is random and unknown to the carrier, ${R_0}^d, \cdots, {R_s}^d \mod n^2$ look random to the carrier and will give him no more information about the plaintext user's profile than the Paillier ciphertexts $Enc(p_0), \cdots Enc(p_s)$. That is, the carrier learns nothing about the user's plaintext profile $X = \{a_1, \cdots, a_s\}$ except its size s. The same holds true for an eavesdropper listening to the communication between the user's device and the carrier during set-up.

At Step 2 of implicit authentication, the carrier only gets the fresh sample Y encrypted under Paillier and randomly re-ordered. Hence, the carrier learns no information on Y, except its size t. At Step 3, the carrier learns $|X \cap Y|$, but not knowing Y, the size $|X \cap Y|$ of the intersection leaks to him no information on X. □

If we define security of implicit authentication as the inability of a dishonest user's device to disrupt the authentication outcome, we can state the following result.

Lemma 3. *A dishonest user's device has no better strategy to alter the outcome of implicit authentication than trying to randomly guess the user's profile.*

Proof. At the end of the set-up protocol, the (still uncompromised) user's keeps no information about the user's profile (Lemma 1). Hence, if the user's device is later compromised and/or behaves dishonestly, it still has no clue on the real user's profile against which its fresh sample is going to be authenticated. Hence, either the user's device provides an honest fresh sample and implicit authentication will be correctly performed, or the user's device provides a random fresh sample with the hope that it matches the user's profile. □

5.7.3 Complexity

Case A

During the set-up protocol, the user's device needs to compute:

- $s+1$ Paillier encryptions for the polynomial coefficients;
- values $r'_0, \cdots, r'_s$; as explained before, this can be done by randomly choosing r'_0, then solving an $s \times s$ generalized Vandermonde system (doable in $O(s^2)$ time using [38]) and finally computing s modular powers to find the $r'_1, \cdots, r'_s$;
- $s+1$ modular powers (raising the R_i values to d).

During the implicit authentication protocol, the user's device needs to compute (Step 2):

- t Paillier encryptions;
- ts modular powers (to compute the Υ_j values);
- t modular powers (to raise R' to $r(j)d$).

Also during the implicit authentication protocol, the carrier needs to compute:

- At Step 1, $s+1$ modular powers (to raise the encrypted polynomial coefficients to θ);
- At Step 3, t Paillier encryptions;
- At Step 3, t modular powers (to raise the Υ_j values to $n\theta$).

Case B

The set-up protocol does not change w.r.t. Case A. In the implicit authentication protocol, the highest complexity occurs when $Y' = E$ and the similarity function l always takes the maximum value in its range, say L. In this case,

$$\sum_{z \in Y'} \ell_z = \sum_{z \in Y'} \sum_{y \in Y} l(z, y) = |E|sL.$$

Hence, in the *worst case* the user's device needs to compute (Step 2'):

- $|E|sL$ Paillier encryptions;
- $|E|sL$ modular powers (to compute the Υ_j values);
- $|E|sL$ modular powers (to raise R' to $r(j)d$).

Also during the implicit authentication protocol, the carrier needs to compute:

- At Step 1, $s + 1$ modular powers (to raise the encrypted polynomial coefficients to θ);
- At Step 3', $|E|sL$ Paillier encryptions;
- At Step 3', $|E|sL$ modular powers (to raise the Υ_j values to $n\theta$).

Note that the above complexity can be reduced by reducing the range of the similarity function $l(\cdot, \cdot)$.

Case C

Case C is analogous to Case A but the sets X and Y, whose intersection is computed, no longer have s and t elements, respectively. According to Section 5.6.3, the maximum value for $|X|$ occurs when all u_i take the maximum value of their range, say, M, in which case X contains tM pairs (i, j). By a similar argument, Y also contains at most tM pairs.

Hence, the *worst-case* complexity for Case C is obtained by performing the corresponding changes in the assessment of Case A. Specifically, during the set-up protocol, the user's device needs to compute:

- $tM + 1$ Paillier encryptions for the polynomial coefficients;

- Solve a Vandermonde system $tM \times tM$ (doable in $O((tM)^2)$ time) and then compute tM modular powers to find the r'_i values;
- Compute $tM + 1$ modular powers (raising the R_i values to d).

During the implicit authentication protocol, the user's device needs to compute (Step 2):

- tM Paillier encryptions;
- t^2M^2 modular powers (to compute the Υ_j values);
- tM modular powers (to raise R' to $r(j)d$).

Also during the implicit authentication protocol, the carrier needs to compute:

- At Step 1, $tM + 1$ modular powers (to raise the encrypted polynomial coefficients to θ);
- At Step 3, tM Paillier encryptions;
- At Step 3, tM modular powers (to raise the Υ_j values to $n\theta$).

Note that the above complexities can be reduced by reducing the range of the numerical values in sets U and V.

5.8 Experimental results

As stated in the previous section, the complexity of our implicit authentication protocol ultimately depends on the sizes of the input sets. In Case A, the sizes of the sets are directly given by the user inputs; in Case B, these sizes are the product of the size of the input sets times the range of the similarity function ℓ; and in Case C, the sizes are given by the size of the original sets times the range of their values. We ran an experiment to test the execution times of our protocol, based on Case A, to which the other two cases can be reduced.

The experiment was implemented in Sage-6.4.1 and run on a Debian 7.7 machine with a 64-bit architecture, an Intel i7 processor and 8GB of physical memory. We instantiated a Paillier cryptosystem with a 1024-bit long n, and the features of preference sets were taken from the integers in the range $[1 \dots 2^{128}]$. The input sets ranged from size 1 to 50, and we took feature sets of the same size to execute the set-up and the authentication protocols.

Table 5.1: Execution times (in seconds) for different input set sizes

# of features	Set-up	Authentication
1	0.89	0.08
5	0.79	0.47
10	1.10	1.05
15	1.83	2.00
20	4.67	3.37
25	11.45	5.40
30	24.65	8.27
35	47.60	12.13
40	84.99	17.30
45	144.81	23.39
50	228.6	31.20

Step 4 of the set-up protocol (Section 5.6.1), in which a system of equations is solved for r'_i for $1 \leq i \leq s$, is the most expensive part of the set-up protocol. As a worst-case setting, we used straightforward Gaussian elimination which takes time $O(s^3)$, although, as mentioned above, specific methods like [38] exist for generalized Vandermonde matrices that can run in $O(s^2)$ (such specific methods could be leveraged in case of smartphones with low computational power). On the other hand, Step 2 of the authentication protocol (Section 5.6.1), computed by the user's device, is easily parallelizable for each feature in the sample set. Since parallelization can be exploited by most of the current smartphones in the market, we also exploited it in our experiment. The results are shown in Table 5.1 (times are in seconds).

Note that the set-up protocol is run only once (actually, maybe once in a while), so it is not time-critical. However, the authentication protocol is to be run at every authentication attempt by the user. For example, if a user implicitly authenticates herself using the pattern of her 20 most visited websites, authentication with our proposal would take 3.37 seconds, which is perfectly acceptable in practice.

5.9 Privacy-preserving implicit authentication from Bloom filters

In this section we present an alternative privacy-preserving implicit authentication protocol leveraging the properties of Bloom filters. This construction is computationally lighter and supports the computation of distances of feature sets of the types A and C (as described in Section 5.4).

We use Bloom filters as described in Chapter 2 to encode the users' profiles so that the server cannot obtain the profile of a user from its Bloom filter representation; this guarantees privacy. The decision whether a user is authenticated or not will be taken by computing the distance between previously recorded profiles and the fresh samples the user provides to the server; authentication will be positive if that distance is below a predefined threshold.

5.9.1 Implicit authentication in cases A and C

Set-up protocol

Let the initial user's profile be $\mathbb{P} = \emptyset$. The aim of the set-up protocol is to begin populating this profile with one or several behavior samples. Let the first sample $\mathcal{S}_0$ be a collection of sets $\mathcal{S}_0 = \{S_{0,i} \mid 1 \leq i \leq p\}$, in which every set $S_{0,i} = \{s_1^{0,i}, \ldots, s_n^{0,i}\}$ is a labeled feature set, for $n \leq N$. The maximum allowed value N, as well as the values k and m (number of hash functions and length of the Bloom filters, respectively) are set by the server (who may be the carrier or the service provider). The weights of each of the feature sets are also set by the server.

The sample $\mathcal{S}_0$ and every subsequent sample $\mathcal{S}_t$ to be added to the user's profile are preprocessed as follows:

- Sets of categorical features $S_{t,i} \in \mathcal{S}_t$ are aggregated as described in Section 5.3.2, by concatenating the label of the feature set to each of the feature values and uniting them into sets $R_{t,j}$, taking into account the weights of each of the features. For example, sets *Applications*: {WhatsApp, Facebook} and *Antennas*: {ANT001, ANT004} can be aggregated into *Applications_Antennas*: {*Applications*:WhatsApp, *Applications*:Facebook, *Antennas*:ANT001, *Antennas*:ANT004}, as long as feature sets *Applications* and *Antennas* weigh the same in the authentication decision.

- Regarding sets of numerical features in $\mathcal{S}_t$, for each $S_{t,i}$ the following set

is built

$$R_{t,i} = \{(j,l) : 1 \leq j \leq n, 1 \leq l \leq s_j^{t,i}\}. \tag{5.4}$$

For example, from $S = \{2, 3, 1\}$, one would build

$$R = \{(1,1), (1,2), (2,1), (2,2), (2,3), (3,1)\}.$$

The sizes of these sets are also required to be less than or equal to N.

The protocol proceeds as follows, using the modified sample $\mathcal{R}_0 = \{R_{0,j} \mid 1 \leq j \leq q\}$:

1. The user initializes a set of Bloom filters $B_{\mathcal{R}_0} = \{B_{R_{0,j}}, 1 \leq j \leq q\}$, each of them of size m, by setting all their bits to 0.

2. Then, the device generates a large random number, say key, and stores it in a secure location (for example, a hardware-backed key storage).

3. The user inserts all elements in $R_{0,j}$ into $B_{R_{0,j}}$, for $1 \leq j \leq q$, that is, for every feature $r_l^{0,j} \in R_{0,j}$, the user computes the index set

   $$I_{r_l^{0,j}} = \{h_0(r_l^{0,j}, \mathsf{key}), \ldots, h_{k-1}(r_l^{0,j}, \mathsf{key})\}$$

   and sets the corresponding bits to 1.

4. Finally, the user sends $B_{\mathcal{R}_0}$ to the server in a confidential manner and deletes $\mathcal{S}_0$, $\mathcal{R}_0$ and $B_{\mathcal{R}_0}$ from the device.

In the last step of the set-up protocol, the user can send $B_{\mathcal{R}_0}$ confidentially to the server by encrypting $B_{\mathcal{R}_0}$ under the server's public key. This protocol can be executed several times within a predefined training period, in order to populate the user's profile $\mathbb{P}$ with additional samples $\mathcal{S}_t$.

Authentication protocol

To authenticate the user, the server needs to compute the distances between the feature sets included in the new behavior sample provided by the user and the reference profile (which may consist of several behavior samples).

Let the user's new collected sample be a list of feature sets $\mathcal{S}_f$. The user preprocesses the sample to obtain the modified sample $\mathcal{R}_f$, in the same way described in the set-up phase, and builds a set of Bloom filters $B_{\mathcal{R}_f}$, also in the same way described in the set-up phase. The authentication protocol proceeds as follows:

1. The user and the server agree on a fresh random secret key K (for example, using Diffie-Hellman key exchange).

2. The user's device sends $B_{\mathcal{R}_f}$ encrypted under K to the server, and deletes $\mathcal{S}_f$, $\mathcal{R}_f$ and $B_{\mathcal{R}_f}$.

3. The server then computes the distances between the protected sample $B_{\mathcal{R}_f}$ and ℓ previously stored samples in the user profile. The feature sets in each of the samples are assumed to be labeled in such a way that only compatible features are compared. The distances are computed as follows:

   - For sets of categorical features, the server computes $|R_{t,j} \cap R_{f,j}|$ and $|R_{t,j} \cup R_{f,j}|$, as described in Section 2.8.1 and using Expression (2.2), and obtains the Jaccard distance $d_{J_{t,f,j}}(R_{t,j}, R_{f,j})$, for $f - \ell < t < f$ and all i.
   - For sets of numerical features, the server computes $|R_{t,j}|$, $|R_{f,j}|$ and $|R_{t,j} \cap R_{f,j}|$, and obtains the distance as

   $$d_{t,f,j} = |R_{t,j}| + |R_{f,j}| - 2|R_{t,j} \cap R_{f,j}| \tag{5.5}$$

   because, using Expression (5.4),

   $$|R_{t,j}| + |R_{f,j}| - 2|R_{t,j} \cap R_{f,j}|$$

   $$= \sum_{l=1}^{n} (\max\{s_l^{t,j}, s_l^{f,j}\} + \min\{s_l^{t,j}, s_l^{f,j}\}) - 2\sum_{l=1}^{n} \min\{s_l^{t,j}, s_l^{f,j}\}$$

   $$= \sum_{l=1}^{n} (\max\{s_l^{t,j}, s_l^{f,j}\} - \min\{s_l^{t,j}, s_l^{f,j}\}) = \sum_{l=1}^{n} |s_l^{t,j} - s_l^{f,j}|.$$

4. By aggregating the distances for all j's according to their weights, the server obtains a vector $\delta = [\delta_{f-\ell-1,f}, \cdots, \delta_{f-1,f}]$ of the distances between the new sample and ℓ past samples in the user profile. This vector is used then to compute a score (*e.g.* the mean and standard deviation of the distances) which is compared to a threshold t to return an authentication decision.

5. On a successful authentication, the server may store $B_{\mathcal{R}_f}$ as additional reference for authentication.

5.10 Privacy and security

As introduced in Section 5.3, a privacy attacker's objective is to learn the behavior of the authenticated users. We have the following two claims related to privacy.

Claim 1 (Privacy at the user's device)**.** *An attacker with access to a user's device cannot obtain past feature samples.*

Justification. The user's device deletes the profiles and the protected profiles after set-up and after every authentication attempt. □

Claim 2 (Privacy at the server)**.** *The server (or an attacker) does not learn anything about the plaintext profiles other than the number of encoded elements and the sizes of the intersections and unions with other protected profiles of the same user.*

Justification. The server does not receive the plaintext feature sets but Bloom filter encodings of such sets. To recover one feature in the plaintext profile, the server, or any attacker with access to the protected profile, needs to find not just a pre-image of the Bloom filter(s) (see Section 2.8.2), but *the right* pre-image corresponding to the feature. If the server finds a spurious pre-image (not corresponding to any element), privacy is not violated.

Now, as described in Section 2.8.3, our instantiation of Bloom filters uses keyed cryptographic hash functions, which, ideally, are resistant to pre-image attacks. These hash functions have much larger domains (as their input includes a long secret key only known to the user's device) and take longer to compute than plain hash functions. This thwarts brute-force attacks attempting to find pre-images by exhaustive search. Hence, finding pre-images (even spurious ones) is not feasible, let alone finding the right pre-images corresponding to inserted features. □

Regarding security against impersonation, we can justify the following claim.

Claim 3 (Security against impersonation)**.** *An impersonator with access to the user's device has no better chance to cheat the system than guessing a sample profile close enough to the reference profile stored at the server, where the impersonator's guess must be made without knowing the reference profile or any previous fresh sample of the legitimate user.*

Justification. By Claim 1, having access to the user's device does not give the impersonator access to past samples. Let us examine his other options,

which consist of attacking the set-up protocol, the authentication protocol or the Bloom filters used in either protocol.

Attacks during set-up. Since the Bloom filters corresponding to samples in the user's profile are registered in a confidential manner at set-up time (see last step of the set-up protocol in Section 5.9.1), the impersonator cannot get hold of any Bloom filter registered by a legitimate user. On the other hand, the set-up protocol cannot be executed a second time after the predefined training period has expired, and so, an attacker cannot replace after set-up the Bloom filters in the user's profile with new Bloom filters of his choice (the only Bloom filters that can be added to the user's profile after set-up are those corresponding to fresh samples *that were successfully authenticated*, see last step of the authentication protocol in Section 5.9.1).

Attacks during authentication. Each fresh sample is encrypted under a different fresh random secret key K agreed upon by the user and the server in the first step of the authentication protocol (see Section 5.9.1). Hence:

- The attacker cannot get hold of any previous plaintext Bloom filter submitted by the legitimate user, and therefore he cannot re-submit this Bloom filter under a fresh key agreed upon between the attacker and the server.

- The attacker cannot replay any eavesdropped encryption of a Bloom filter under a previous K' agreed upon between the legitimate user and the server, because a fresh K must be agreed upon each time the implicit authentication protocol is run.

Attacks to Bloom filters. Finding a second pre-image for unknown legitimate (B_S, S), where S is a feature set and B_S is a Bloom filter containing the elements of S, is not easier than finding a second pre-image for known (B_S, S); the latter is difficult in our instantiation of Bloom filters based on keyed cryptographic hash functions. Similarly, finding collisions S, S' for unknown legitimate (B_S, S) is not easier than finding collisions for known (B_S, S); the latter is difficult in our instantiation of Bloom filters based on keyed cryptographic hash functions. Hence, the only strategy to pass the authentication protocol that remains to an attacker having access to the user's device is to *guess* a fresh sample $\mathcal{S}_f$ such that its corresponding Bloom filters $B_{\mathcal{R}_f}$ are close to some of the Bloom filters stored by the server for the user to be impersonated. □

5.11 Experimental analysis

To test the applicability of our mechanism, we implemented our implicit authentication protocol in Python. Our choice for the hash functions in the Bloom filters is $h_i(x, \mathsf{key}) = \mathsf{SHA\text{-}512}(x) + i\mathsf{HMAC}(x, \mathsf{key}) \mod m$, with i ranging from 1 to k. This construction follows the guidelines of [65] and is assumed to be resistant against preimage and second preimage attacks; however, new security requirements and attacks may require updating the above choice of hash functions (this cautionary note is similar to the caveats about hash functions in the context of digital signatures). The system in which the tests were performed is an Ubuntu x64 14.04 LTS running on an Intel i7-2600 at 3.4 GHz and with 16GB DDR3 1333 Mhz.

5.11.1 Speed test for categorical features

In a first test, we checked the speed of the set-up and implicit authentication protocols for different values of n, the size of the feature sets. Parameters m and k were set as per Equations (2.3). A user's profile containing a single feature set was considered and the results are shown in Figure 5.5. The running time of the set-up protocol does not include the encryption of B_R. Also, the running time of the implicit authentication protocol does not include the execution of the protocol to agree on K. The reason to exclude such cryptographic components is that we use them as a black box: they are not part of the core of our proposal and their running time depends very much on the precise cryptographic algorithms and implementations used.

It may be surprising that the running times of both protocols are so similar. The explanation is that both protocols perform very similar tasks. As described in the previous section, the set-up protocol builds a Bloom filter containing the features in the set and sends it to the server. On the other hand, the implicit authentication protocol builds a Bloom filter with the features in a fresh sample and sends it to the server, who compares it with the reference profile received at set-up time. It turns out that the time needed to compare two Bloom filters is negligible (less than 1 ms) with respect to the time needed to build a Bloom filter (23.4 s for $n = 1,000,000$), so that the latter dominates the total running time of the implicit authentication protocol.

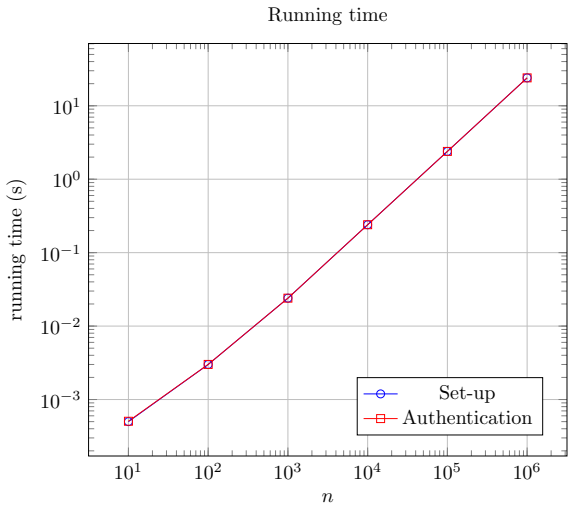

Figure 5.5: Categorical features. Running times for different values of the feature set size n. The user profile is assumed to contain a single feature set.

5.11.2 Accuracy test for categorical features

In this second test, we measured the loss of accuracy introduced by our mechanism in the case of independent categorical features. For such a purpose, we generated $5,000$ pairs of feature sets of size $n = 50$ with independent categorical features. The first feature set of each pair was taken as the user's profile submitted at set-up time. The second one was taken as a fresh sample submitted at authentication time. Each of the fresh samples was modified by randomly changing up to 50% of their features. Next, with a threshold $t = 0.3$, we classified each of the pairs by computing the Jaccard distance of the pair and tagging it as *accepted* if its distance was below the threshold or *rejected* otherwise. Then, we ran our protocol for all pairs with different values of m and k and we counted how many pairs were misclassified (that is, *accepted* pairs that did not pass authentication plus *rejected* pairs that passed it). The results are shown in Figure 5.6.

The vertical dashed line at approximately $m = 2^{9.5}$ is the optimal value for m as per Expressions (2.3), using $N = n = 50$ and $\rho = 0.001$. We can see that for values of m greater than or equal to the optimal value, the error rate falls

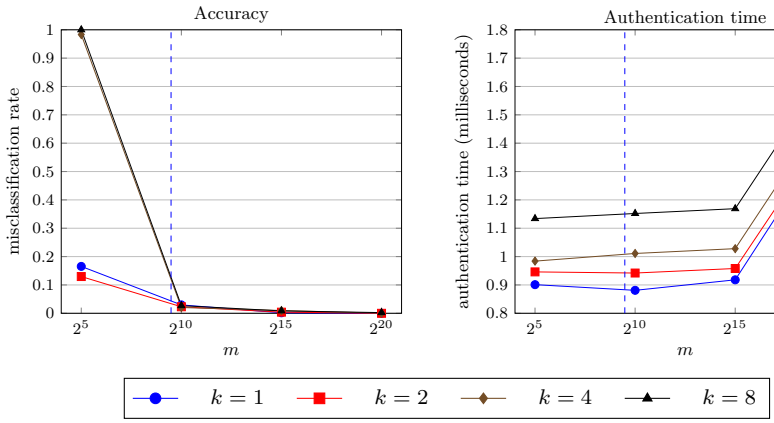

Figure 5.6: Categorical features. Accuracy and running times for different values of m and k. Results were obtained as the average of 5,000 pairs of feature sets, each containing $n = 50$ features.

below 5%; in fact, it approaches 0% for m close to 2^{20}. Note that for $m = 2^5$ and $k = 4$ or $k = 8$, the error rate is 1. This is because for values of m below the optimal value, the Bloom filter is quickly filled with 1's. When all its m bits become 1's, by Expression (2.2) the Bloom filter contains ∞ elements ($-\ln 0$); in other words, any element passes the membership test. Hence, a large filter size m is recommended, and the implicit authentication time plot shows that sizes as large as $m = 2^{20}$ are quite manageable in terms of time. Regarding storage and communication, large m values are not problematic either, because Bloom filters can be easily stored and sent in lossless compressed form.

5.11.3 Accuracy and speed test for numerical features

In this test, we checked the accuracy of our mechanism in the case of independent numerical features. We followed an approach similar to the one in the previous test. We generated a reference profile consisting of a single feature set S_0 consisting of $n = 50$ percentage values adding to 100, that is, a normalized histogram. The features in S_0 might represent the user's 50 most visited websites with their relative frequencies. From S_0, we generated additional feature sets S_i for $1 \leq i \leq 100$, by modifying each of the 50 features of S_0 randomly and re-normalizing. Feature set S_0 was taken as the user's profile submitted as

set-up time. Sets $S_1, \ldots, S_{100}$ were taken as fresh samples.

We computed the difference between two samples as $d(S_0, S_i) = \sum_{j=1}^{50} |s_j^0 - s_j^i|$. The average introduced distance $d(S_0, S_i)$ for $1 \leq i \leq 100$ was 5.43. As our method cannot deal with real numbers, we made a transformation by multiplying by 100 and rounding each feature values. This procedure introduced an error of 0.46%. Note that multiplying feature values by 100 to eliminate the decimal positions increases the size of the sets R_0 and R_i, respectively built during set-up and authentication according to Expression (5.4), because the maximum values for s_j^0 and s_j^i increase by a factor of 100.

We set $m = 2^{20}$ and $k = 4$. By using our implicit authentication method, the average error introduced for $1 \leq i \leq 100$ was 0.83%.

Beyond measuring the accuracy, we also measured the time taken by the set-up and the implicit authentication protocols. Since we are in the independent numerical feature case, the set-up protocol consisted of building the set R_0 from the user's profile feature set S_0, and building the corresponding Bloom filter (see Section 5.9.1). The mean time in this case was 133 ms. The implicit authentication protocol consisted of building the set R_i from a fresh sample profile S_i, for some $i \in \{1, \ldots, 100\}$, then computing the sizes $|R_0|$, $|R_i|$ and $|R_0 \cap R_i|$ and finally computing the distance from these sizes as per Expression (5.5). The mean time of the authentication protocol was 133.3 ms (very similar to the mean time of the set-up protocol).

5.11.4 Tests on the GCU dataset

Finally, we tested our authentication mechanism with the GCU Dataset Version 1 [63]. The GCU Dataset contains sensor data from 7 Android users. The data consist of WiFi networks, cell towers, application use, light and sound levels, etc.

In this experiment, we used the application usage data to authenticate users. Since these data are categorical and independent, we used the Jaccard similarity coefficient to test the users. The experiment began by choosing one of the 7 users at random, and building a reference user profile by storing 50 samples of the chosen user. Then, we made $1,000,000$ authentication attempts, by choosing sample readings at random from all 7 users. We authenticated a user when the average similarity, computed as the average similarity of the random sample against the 50 reference samples, was above a pre-specified threshold, taking into account the standard error of the computed average.

In Figure 5.7, we show the performance metrics of this authentication mech-

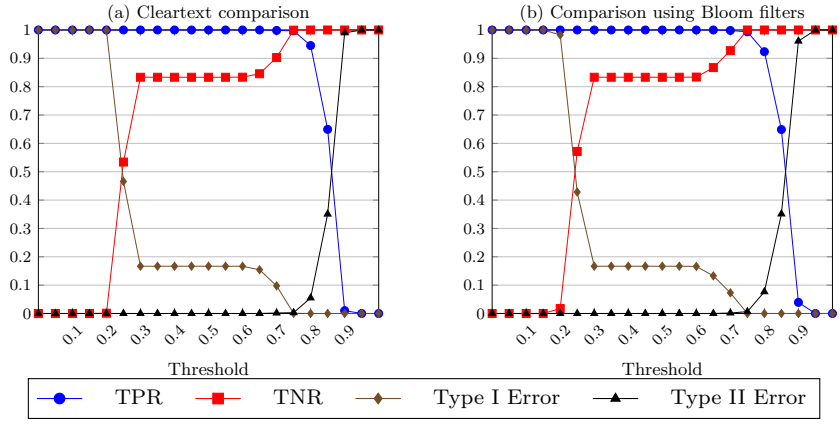

Figure 5.7: Performance indicators of the implicit authentication mechanism on the GCU dataset. TPR stands for true positive rate (correct authentication) and TNR for true negative rate (correct non-authentication). Type I Error stands for false positive rate and Type II for false negative rate.

anism for the GCU dataset, including false positive and negative rates, for increasing values of the threshold. Figure 5.7a shows the results of the experiment when computing the distances with cleartext samples, while Figure 5.7b shows the results for comparisons using our proposed mechanism based on Bloom filters. Note that the results of both approaches are nearly identical, showing that the use of Bloom filters to protect the user profiles does not significantly affect the authentication accuracy with respect to comparing the profiles in the clear.

An additional positive side effect of using Bloom filters is that the encoded samples are significantly smaller than the average cleartext sample.

5.12 Summary

We have presented two privacy-preserving implicit authentication mechanisms based on the private computation of the size of the intersections, using two different primitives, which grant the two protocols similar functionalities but slightly different properties.

To the best of our knowledge, our privacy-preserving implicit authentication mechanism based on the homomorphic properties of the Paillier cryptosystem

is the second system in the literature after the one in [82]. The advantages of this proposal with respect to [82] are:

- The carrier only needs to store the user's profile encrypted under *one* cryptosystem, namely Paillier's.
- Dishonest behavior or compromise at the user's device after the initial set-up stage neither compromises the privacy of the user's profile nor affects the security of authentication.
- Our proposal is not restricted to numerical features, but can deal also with all sorts of categorical features.
- In case of numerical or categorical ordinal features, our proposal does not disclose how the fresh sample is ordered with respect to the feature values in the stored user's profile.

For binary or independent nominal features, the complexity of our proposal is quite low (quadratic in the number of values in the user's profile). For correlated categorical feature values, the complexity is higher, but it can be reduced by decreasing the range of the similarity function used. Finally, in the case of numerical values, the complexity is also higher than in the binary/independent nominal case, but it can be reduced by decreasing the range of the numerical feature values.

Our second proposal, based on Bloom filters, improves significantly the computational complexity of the implicit authentication protocol, while keeping comparable properties to the first one. Our efficiency improvement, however, comes at the cost of losing the semantic security provided by the previous protocol. In an extreme scenario, such a loss might impact on the privacy of our solution.

Chapter 6

Conclusions and future work

6.1 Conclusions

In this thesis we have contributed privacy-aware schemes for group discounts, loyalty programs and implicit authentication that maintain, to a very great extent, the functionalities of the usual non-private counterparts. While privacy preservation comes at some loss of performance, in some cases such a loss is only marginal (for example, the implicit authentication based on Bloom filters is nearly as efficient as the non-private implicit authentication). In any case, the performance loss is quantified and limited, and the proposed privacy-aware protocols have been shown to be usable in practice. We believe that our contributions exemplify that e-commerce services that currently invade the privacy of the users can be made privacy-preserving without loss of functionality and with acceptable performance.

Specifically, our contributions are the following:

1. A group size accreditation method that preserves anonymity of the members of the groups. The anonymity provided by the scheme is configurable. The method rests on two building blocks:

   (a) A new parameterized key management scheme for identity-based signatures that allows setting the anonymity level of users by providing

them with multiple keys that are shared by many other users, but that are extracted from a unique identity.

(b) A novel IBDT signature scheme based on asymmetric bilinear pairings, that combines the properties of identity-based and threshold signature schemes. Signatures produced with this scheme reveal only the public keys of the group members, which are called identities, and the size of the signing group. The signature scheme is efficient, and the sizes of the signatures are constant.

2. A privacy-preserving loyalty program protocol suite, whereby vendors can issue and verify loyalty points, and customers can maintain their anonymity and configure the level of generalization for their purchase receipts before submitting them for additional loyalty points. This allows vendors to still carry out client profiling in a privacy-aware way. This protocol suite combines the following techniques:

   (a) A new construction for anonymous (untransferable) tokens with controlled linkability based on partially blind signatures and zero-knowledge proofs. The construction allows issuing and verifying tokens, while the verifier cannot link tokens to a specific user or between concrete executions of the issuance and verification procedures, unless such a linkage is authorized by the user. Moreover, if a hardware-based keystore is available, the tokens can be made untransferable, so that only users who originally received the tokens can submit them.

   (b) Generalization techniques to select the level of anonymization of purchase receipts.

3. A mechanism to compute the distance between user profiles (expressed as feature sets of different data types) based on the size of the intersection of the feature sets.

4. A privacy-preserving implicit authentication mechanism using the homomorphic properties of the Paillier cryptosystem, that protects the privacy of the sensitive data in the user's profile and ensures that the server does not learn anything about the user's behavior.

5. A second privacy-preserving implicit authentication with similar functionalities and higher speed compared to the previous one, based on the intersection of Bloom filters. While this mechanism provides slightly less protection than the previous one, its substantially better performance makes it ideal for implementation in existing authentication suites.

6.2 Publications

Next, we enumerate the publications that back the contents of this thesis:

1. Oriol Farràs, Josep Domingo-Ferrer and Alberto Blanco-Justicia, "Privacy-preserving trust management mechanisms from private matching schemes", in *Lecture Notes in Computer Science*, vol. 8247, pp. 390-398. Vol. *Data Privacy Management and Autonomous Spontaneous Security - 8th International Workshop, DPM 2013, and 6th International Workshop, SETOP 2013*, Springer, 2014.

2. Alberto Blanco-Justicia, Josep Domingo-Ferrer, Oriol Farràs and David Sánchez, "Distance computation between two private preference functions", in *IFIP SEC 2014-Intl. Information Security and Privacy Conference*, IFIP AICT 428, pp. 460-470. Springer, 2014. Acceptance rate 27/151.

3. Josep Domingo-Ferrer and Alberto Blanco-Justicia, "Group discounts compatible with buyer privacy", in *Lecture Notes in Computer Science*, vol. 8872, pp. 47-57. Vol. *DPM/SETOP/QASA 2014*, Springer, 2015.

4. Alberto Blanco-Justicia and Josep Domingo-Ferrer, "Privacy-preserving loyalty programs", in *Lecture Notes in Computer Science*, vol. 8872, pp. 133-146. Vol. *DPM/SETOP/QASA 2014*, Springer, 2015.

5. Josep Domingo-Ferrer, Qianhong Wu and Alberto Blanco, "Flexible and robust privacy-preserving implicit authentication", in *IFIP SEC 2015-Intl. Information Security and Privacy Conference*, IFIP AICT 455, pp. 18-34. Springer, 2015. Acceptance rate 42/232.

6. Alberto Blanco-Justicia and Josep Domingo-Ferrer, "Privacy-aware loyalty programs", *Computer Communications*, vol. 82, pp. 83-94, 2016.

7. Josep Domingo-Ferrer, Alberto Blanco-Justicia, and Carla Ràfols, "Dynamic group size accreditation and group discounts preserving anonymity", *International Journal of Information Security* (revision submitted after first review results).

8. Alberto Blanco-Justicia and Josep Domingo-Ferrer. "Efficient privacy-preserving implicit authentication", *Information Sciences* (submitted).

6.3 Future work

Concerning our privacy-preserving group size accreditation method, we plan to integrate it with a broad range of anonymous or near-anonymous payment systems, in view of facilitating its effective adoption for group discounts. Beyond group discounts, we will also explore additional applications of privacy-aware group size accreditation.

Our anonymous (untransferable) tokens with controlled linkability are an interesting primitive to support not only loyalty programs, but other applications which involve, for example, tickets, receipts or temporary authorizations with anonymity requirements. We plan to work further on this construction, to make it more efficient, for example by reducing the number of messages exchanged in the verification protocol for untransferable tokens.

Regarding implicit authentication, while our second protocol is a great improvement with respect to our first proposal in terms of performance, it cannot offer the semantic security of the first protocol. We plan to address this problem in future work, by considering the use of oblivious transfer protocols and homomorphic encryption (such as Goldwasser-Micali). Another line of future research relates to finding ways of using Bloom filters to deal with correlated features in profiles (case B in Section 5.4.2), that is, features that are not independent of each other (for example, if the feature values are the IDs of cell towers or Internet access points seen by the device, nearby cell towers/access points are more similar to each other than distant cell towers/access points). This requires being able to deal with multisets, which is not doable with standard Bloom filters.

Bibliography

[1] "Consumers reveal privacy concerns with loyalty programs," *Convenience Store Decisions*, 2014. http://www.cstoredecisions.com/2013/10/28/consumers-reveal-privacy-concerns-loyalty-programs/

[2] "One million fewer customer visits a week at Tesco," *The Guardian*, 2014. http://www.theguardian.com/business/2014/jun/03/tesco-morrisons-sales-fall-further-aldi-lidl

[3] "Tor Project: Anonymity Online," accessed February 8, 2017. www.torproject.org

[4] "White Paper: An Overview of the Samsung KNOX Platform," accessed February 8, 2017. http://www.samsung.com/uk/business-images/insights/2015/An_Overview_of_the_Samsung_KNOX_Platform_V1.11-0.pdf

[5] M. Abdalla, F. Benhamouda and D. Pointcheval, "Disjunctions for hash proof systems: new constructions and applications," in *Advances in Cryptology - EUROCRYPT 2015*, Part II LNCS 9057, pp. 69–100, Springer, 2015.

[6] M. Abe and E. Fujisaki, "How to date blind signatures," in *Advances in Cryptology–ASIACRYPT 1996*, LNCS 1163, pp. 244–251, Springer, 2002.

[7] M. Abe and T. Okamoto, "Provably secure partially blind signatures," in *Advances in Cryptology–CRYPTO 2000*, LNCS 1880, pp. 271–286, Springer, 2000.

[8] F. Abel, Q. Gao, G.J. Houben and K. Tao, "Semantic enrichment of Twitter posts for user profile construction on the social web", in *ESWC 2011*, pp. 375–389, 2011.

[9] N. Attrapadung, B. Libert and E. de Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *Public Key Cryptography—PKC 2011*, LNCS 6571, pp. 90–108, Springer, 2011.

[10] Y. Aumann and Y. Lindell, "Security against covert adversaries: efficient protocols for realistic adversaries," in *Journal of Cryptology*, 23(2):281–343, 2010.

[11] M. Bellare, H. Shi and C. Zang, "Foundations of group signatures: the case of dynamic groups," in *CT - RSA 2005*, LNCS 3376, pp. 136–153, Springer, 2005.

[12] G. R. Blakley, "Safeguarding cryptographic keys," in *Proceedings of the National Computer Conference*, pp. 313–317, New York: AFIPS Press, 1979.

[13] A. Blanco and J. Domingo-Ferrer, "Privacy-preserving loyalty programs," in *9th Intl. Workshop on Data Privacy Management–DPM 2014*, LNCS 8872, pp. 133-146, Springer, 2015.

[14] A. Blanco-Justicia and J. Domingo-Ferrer, "Privacy-aware loyalty programs," in *Computer Communications*, vol. 82, pp. 83-94, Elsevier, 2016.

[15] A. Blanco-Justicia, J. Domingo-Ferrer, O. Farràs and D. Sánchez, "Distance computation between two private preference functions," in *Proc. of 29th IFIP TC 11 Intl. Conference (SEC 2014)*, pp. 460–470, Springer, 2014.

[16] M. Blanton and E. Aguiar, "Private and oblivious set and multiset operations," in *ASIACCS 2012*, pp. 40-41, ACM, 2012.

[17] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM* 13(7):422–426, ACM, 1970.

[18] Bluetooth SIG, *Specification of the Bluetooth System*, 2013. Available in `https://www.bluetooth.org/en-us/specification/adopted-specifications`.

[19] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme," *Public Key Cryptography–PKC 2003*, LNCS 2567, pp. 31–46, Springer, 2003.

[20] A. Boldyreva, N. Chenette and A. O'Neill, "Order-preserving symmetric encryption", in *Advances in Cryptology–EUROCRYPT 2009*, LNCS 5479, Springer, pp. 224-241, 2009.

[21] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *Advances in Cryptology–EUROCRYPT 2004*, LNCS 3027, pp. 223–238, Springer, 2004.

[22] D. Boneh, X. Boyen and E.-J. Goh, "Hierarchical identity-based encryption with constant size ciphertext," in *Advances in Cryptology–EUROCRYPT 2005*, LNCS 3494, pp. 440–456, Springer, 2005.

[23] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology–CRYPTO 2001*, LNCS 2139, pp. 213–229, Springer, 2001.

[24] D. Boneh, C. Gentry and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Advances in Cryptology–CRYPTO 2005*, LNCS 3621, pp. 258–275, Springer, 2005.

[25] D. Boneh and M. Hamburg, "Generalized identity-based and broadcast encryption schemes," in *Advances in Cryptology–ASIACRYPT 2008*, LNCS 5350, pp. 455–470, Springer, 2008.

[26] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Weil pairing," in *Advances in Cryptology–ASIACRYPT 2001*, LNCS 2248, pp. 514–532, Springer, 2001.

[27] P. Bose, H. Gua, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid and Y. Tang, "On the false-positive rate of Bloom filters," *Information Processing Letters* 108.4, pp. 210–213, 2008.

[28] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology–EUROCRYPT 2001*, LNCS 2045, pp. 93–118, Springer, 2001.

[29] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology–CRYPTO 2004*, LNCS 3152, pp. 56–72, Springer, 2004.

[30] CBC News Canada, "Man charged for driving with 2 mannequins in HOV lane," `http://www.cbc.ca/news/canada/toronto/man-charged-for-driving-with-2-mannequins-in-hov-lane-1.3143701`. Accessed February 8, 2017.

[31] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology–CRYPTO 82*, pp. 199–203, Springer, 1983.

[32] D. Chaum, A. Fiat and M. Naor, "Untraceable electronic cash," in *Advances in Cryptology–CRYPTO 88*, LNCS 403, pp. 319–327, Springer, 1990.

[33] L. Chen, P. Morrissey and N. P. Smart, "Pairings in trusted computing," in *Pairing 2008*, LNCS 5209, pp. 1–17, Springer, 2008.

[34] R. Cramer, I. Damgård and B. Schoenmakers. "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology–CRYPTO 1994*, LNCS 839, pp. 174–187, Springer, 1994.

[35] J.P. Davis, J.M.A McNamara and J.D. Rector, "Devices, systems and methods for identifying and/or billing an individual in a vehicle," US Patent US8280791 B2. Date filed: Dec. 8, 2009.

[36] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications*, pp. 850–855, IEEE, 2011.

[37] E. De Cristofaro, P. Gasti and G. Tsudik, "Fast and private computation of cardinality of set intersection and union," in *CANS 2012*, pp. 218-231, Springer, 2012.

[38] J. Demmel and P. Koev, "The accurate and efficient solution of a totally positive generalized Vandermonde linear system," in *SIAM Journal on Matrix Analysis and Applications*, 27.1, pp. 142-152, 2005.

[39] DG CONNECT, "Flash Eurobarometer 443 – e-Privacy," 2016. `http://ec.europa.eu/COMMFrontOffice/publicopinion/index.cfm/ResultDoc/download/DocumentKy/76377`

[40] Y. Dodis, A. Kiayias, A. Nicolosi and V. Shoup, "Anonymous identification in ad-hoc Groups," in *Advances in Cryptology–EUROCRYPT 2004*, LNCS 3029, pp. 609–627, Springer, 2004.

[41] J. Domingo-Ferrer and A. Blanco-Justicia, "Group discounts compatible with buyer privacy," in *9th Intl. Workshop on Data Privacy Management-DPM 2014*, LNCS 8872, pp. 47-57, Springer, 2015.

[42] J. Domingo-Ferrer, C. Ràfols and J. Aragonès-Vilella, *Method and System for Customized Contactless Toll Collection in Carpool Lanes* (in Spanish "Método y sistema de cobro sin contacto, por el uso de una vía, para vehículos de alta ocupación"), Spanish patent P201200215. Date filed: February 28, 2012.

[43] J. Domingo-Ferrer, Q. Wu and A. Blanco-Justicia, "Flexible and robust privacy-preserving implicit authentication," in *Proc. of the 30th IFIP TC 11 Intl. Conference (SEC 2015)* pp. 18–34, Springer, 2015.

[44] C. Dong, L. Chen and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, pp. 789–800, ACM, 2013.

[45] C. Dunn, "Loyalty programs and privacy issues: do you need to worry about providing personal information?" *Disney Family*, accessed February 8, 2017. `family.go.com`

[46] O. Farràs, J. Domingo-Ferrer and A. Blanco-Justicia, "Privacy-preserving trust management mechanisms from private matching schemes," in *Lecture Notes in Computer Science*, vol. 8247, pp. 390-398. Vol. *Data Privacy Management and Autonomous Spontaneous Security - 8th International Workshop, DPM 2013, and 6th International Workshop, SETOP 2013*, Springer, 2014.

[47] Federal Trade Commission, *Data Brokers: A Call for Transparency and Accountability*, May 2014.

[48] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in *Conference on the Theory and Application of Cryptographic Techniques*, pp. 186–194, Springer, 1986.

[49] J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology – EUROCRYPT 2004*, LNCS 3027, pp. 1–19, Springer, 2004.

[50] S. D. Galbraith, K. G. Paterson, N. P Smart, "Pairings for cryptographers," *Discrete Applied Mathematics* 156(16):3113–3121, Elsevier, 2008.

[51] T. Gerbet, A. Kumar and C. Lauradoux, "The power of evil choices in Bloom filters," in *45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2015)*, pp. 101–112, IEEE, 2015.

[52] S. Goldwasser, S. Micali and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on computing*, 18(1):186–208, 1989.

[53] A. González, A.Hevia and C. Ràfols, "QA-NIZK arguments in asymmetric groups: new tools and new constructions," in *Advances in Cryptology–ASIACRYPT 2015*, Part I LNCS 9452, pp. 605-629, Springer, 2015.

[54] V. Goyal, O. Pandey, A. Sahai and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *ACM CCS 2006*, pp. 89–98, ACM Press, 2006.

[55] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Advances in Cryptology–EUROCRYPT 2008*, LNCS 4965, pp. 415–432, Springer, 2008.

[56] J. Herranz, F. Laguillaumie, B. Libert and C. Ràfols, "Short attribute-based signatures for threshold predicates," in *Topics in Cryptology–CT-RSA 2012*, LNCS 7178, pp. 51–67, Springer, 2012.

[57] S. Hohenberger and S. Weis, "Honest-verifier private disjointness testing without random oracles," in *PET 2006*, LNCS 4258, pp. 277–294, Springer, 2006.

[58] M. Jakobsson, E. Shi, P. Golle and R. Chow, "Implicit authentication for mobile devices," in *Proc. of the 4th USENIX Conf. on Hot Topics in Security.* USENIX Association, 2009.

[59] R.C. Johnson, *eDropship: Methods and Systems for Anonymous eCommerce Shipment*, US Patent 7,853,481, 2010.

[60] C.S. Jutla and A. Roy, "Shorter quasi-adaptive NIZK proofs for linear subspaces", in *ASIACRYPT 2013*, LNCS 8269, pp. 1-20, Springer, 2013.

[61] C.S. Jutla and A. Roy, "Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces", in *Advances in Cryptology–Crypto 2014*, LNCS 8617, pp. 295-312, Springer, 2014.

[62] M. Kantarcioglu, R. Nix and J. Vaidya, "An efficient approximate protocol for privacy-preserving association rule mining," in *Advances in Knowledge Discovery and Data Mining*, LNCS 5476, pp. 515–524, Springer, 2009.

[63] H. G. Kayacik, M. Just, L. Baillie, D. Aspinall, N. Micallef, "Data driven authentication: on the effectiveness of user behaviour modelling with mobile device sensors," *arXiv preprint arXiv:1410.7743*.

[64] F. Kerschbaum. "Outsourced private set intersection using homomorphic encryption," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pp. 85–86, ACM, 2012.

[65] A. Kirsch and M. Mitzenmacher. "Less hashing, same performance: building a better Bloom filter," in *Algorithms–ESA 2006*, LNCS 4168, pp. 456–467, Springer, 2006.

[66] L. Kissner and D. X. Song, "Privacy-preserving set operations," in *Advances in Cryptology-CRYPTO 2005*, LNCS 3621, pp. 241–257, Springer, 2005.

[67] B. Libert, S. Ling, F. Mouhartem, K. Nguyen and H. Wang, "Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions," in *IACR Cryptology ePrint Archive*, 2016.

[68] B. Libert, T. Peters, M. Joye and M. Yung, "Non-malleability from malleability: simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures," in *Advances in Cryptology–EUROCRYPT 2014*, LNCS 8441, pp. 514-532, Springer, 2014.

[69] B. Lynn, *On the Implementation of Pairing-based Cryptosystems*, Doctoral dissertation, Stanford University, 2007.

[70] U. Maurer, "Unifying zero-knowledge proofs of knowledge," in *Progress in Cryptology–AFRICACRYPT 2009*, LNCS 5580, pp. 272–286, Springer, 2009.

[71] I. Miers, C. Garman, M. Green and A. D. Rubin, "Zerocoin: anonymous distributed e-cash from Bitcoin," in *2013 IEEE Symposium on Security and Privacy*, pp. 397–411, IEEE, 2013.

[72] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Consulted*, vol. 1, 2008. Available in `http://www.bitcoin.org/bitcoin.pdf`.

[73] C.A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proceedings of the 8th ACM conference on Computer and Communications Security*, pp. 116–125, ACM, 2001.

[74] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology-EUROCRYPT 1999*, LNCS 1592, pp. 223-238, Springer, 1999.

[75] PatientsLikeMe, `http://www.patientslikeme.com`

[76] Paysafecard, `http://paysafecard.com`, checked Dec. 1, 2014.

[77] B. Pinkas, T. Schneider and M. Zohner. "Faster private set intersection based on OT extension." in *23rd USENIX Security Symposium (USENIX Security 14)*. pp. 797–812, 2014.

[78] A. Pratt, "Loyalty cards vs. privacy concerns," in *Infosec ISLAND*, 2011. `http://www.infosecisland.com/blogview/13729-Loyalty-Cards-vs-Privacy-Concerns.html`

[79] R.A. Popa, A.J. Blumberg, H. Balakrishnan and F.H. Li, "Privacy and accountability for location-based aggregate statistics," in *Proceedings of the 18th ACM conference on Computer and Communications Security*, pp. 653–666, ACM, 2011.

[80] C. Ràfols, "Stretching Groth-Sahai proofs: NIZK proofs of partial satisfiability," in *TCC 2015*, LNCS 9015, pp. 247–276, Springer, 2015.

[81] F. Reid and M. Harrigan, "An analysis of anonymity in the Bitcoin system," in *Security and Privacy in Social Networks*, eds. Y. Altshuler *et al.*, pp. 197–223, Springer, 2013.

[82] N. A. Safa, R. Safavi-Naini and S. F. Shahandashti, "Privacy-preserving implicit authentication," in *IFIP SEC 2014-Intl. Information Security and Privacy Conference*, IFIP AICT 428, pp. 471–484, 2014.

[83] D. Sánchez, M. Batet, D. Isern and A. Valls, "Ontology-based semantic similarity: A new feature-based approach", *Expert Systems with Applications*, 39(9):7718-7728, 2012.

[84] R. Schnell, T. Bachteler and J. Reiher. "Privacy-preserving record linkage using Bloom filters." *BMC Medical Informatics and Decision Making* 9(1):41, 2009.

[85] C.P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology–CRYPTO 89*, LNCS 435, pp. 239–252, Springer, 1990.

[86] A. Shamir, "How to share a secret," *Communications of the ACM*, 22:612–613, 1979.

[87] A. Shamir, "Identity based cryptosystems and signature schemes," in *Advances in Cryptology–CRYPTO 1984*, LNCS 196, pp. 47–53, Springer, 1985.

[88] S. J. Swamidass and P. Baldi. "Mathematical correction for fingerprint similarity measures to improve chemical retrieval". *Journal of Chemical Information and Modeling* 47(3):952–964, 2007.

[89] A.M.K. Tarabia, "Analysis of two queues in parallel with jockeying and restricted capacities," in *Applied Mathematical Modelling* 32(5):802-810, 2008.

[90] "TrustZone - ARM," accessed February 8, 2017. `www.arm.com/products/processors/technologies/trustzone/`

[91] B. Tuttle, "A disloyalty movement? Supermarkets and customers drop loyalty card programs," *TIME*, 2013. `business.time.com/2013/07/11/a-disloyalty-movement-supermarkets-and-customers-drop-loyalty-card-programs/`

[92] J. Vaidya and C. Clifton, "Secure set intersection cardinality with application to association rule mining," *Journal of Computer Security*, 13(4):593-622, 2005.

[93] A. Viejo, D. Sánchez and J. Castellà-Roca, "Preventing automatic user profiling in Web 2.0 applications," *Knowledge-based Systems*, 36:191-205, 2012.

[94] Walmart Open API. `https://developer.walmartlabs.com/`

[95] B. Waters, "Efficient identity-based encryption without random oracles," in *Advances in Cryptology-EUROCRYPT 2005*, LNCS 3494, pp. 114–127, Springer, 2005.

[96] A.C.-C. Yao, "How to generate and exchange secrets", in *27th Annual Symposium on Foundations of Computer Science-FOCS 1986*, pp. 162–167, IEEE, 1986.

[97] F. Zhang, R. Safavi-Naini and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," in *Public Key Cryptography–PKC 2004*, LNCS 2947, pp. 277–290, Springer, 2004.

[98] F. Zhang, R. Safavi-Naini and W. Susilo, "Efficient verifiably encrypted signature and partially blind signature from bilinear pairings," in *Progress in Cryptology–INDOCRYPT 2003*, LNCS 2904, pp. 191–204, Springer, 2003.

[99] K. Zoltan and S. Johann, "Semantic analysis of microposts for efficient people to people interactions", in *10th Roedunet International Conference-RoEduNet'11*, pp. 1–4, IEEE, 2011.

UNIVERSITAT
ROVIRA i VIRGILI