International Journal of Information Security manuscript No.
(will be inserted by the editor)

Dynamic Group Size Accreditation and Group Discounts Preserving Anonymity

Josep Domingo-Ferrer · Alberto Blanco-Justicia · Carla Ràfols

Received: date / Accepted: date

Abstract Group discounts are used by vendors and authorities to encourage certain behaviors. For example, group discounts can be applied to highway tolls to encourage ride sharing, or by museum managers to ensure a minimum number of visitors and plan guided tours more efficiently. We show how group discounts can be offered without forcing customers to surrender their anonymity, as long as customers are equipped with some form of autonomous computing device (*e.g.* smartphone, tablet or computer). Specifically, we present a protocol suite for privacy-aware group discounts that allows a group of customers to prove how many they are without disclosing their identities. The group does not need to be a stable one, but can have been formed on the fly. Coupled with an anonymous payment system, this makes group discounts compatible with buyer privacy (in this case, buyer anonymity). We present a detailed complexity analysis, we give simulation results and we report on a pilot implementation.

Keywords Buyer privacy · Group size accreditation · Group discounts · Digital signatures · Smartphones · Short-range communications

J. Domingo-Ferrer · A. Blanco-Justicia
UNESCO Chair in Data Privacy,
Department of Computer Engineering and Mathematics,
Av. Països Catalans 26, 43007 Tarragona, Catalonia
Tel.: +34 977 558 109
E-mail: {josep.domingo,alberto.blanco}@urv.cat

C. Ràfols
Universitat Pompeu Fabra,
Department of Information and Communications Technologies,
Tànger 122-140, 08018 Barcelona, Catalonia
Tel.: +34 935 421 858
E-mail: carla.rafols@upf.edu

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 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.

In this paper, we propose 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 [24], but adapted to the asymmetric pairing setting.

The rest of the paper is structured as follows. Section 2 presents related work on group size accreditation systems and on advanced cryptographic signatures. Section 3 presents cryptographic background, a review of relevant anonymous payment systems and a review of relevant communications technologies (communications should be short-range in applications where one wants to check that the group members are physically together). Section 4 describes the new IBDT primitive, states its security model, instantiates the primitive, gives a theorem on its security and discusses the choice of parameters for implementation. Section 5 introduces the key management scheme used in our proposal. Section 6 describes our group size accreditation method, including the required entities and protocols. The security and the privacy of our method are analyzed in Section 7. In Section 8, we give a complexity estimation of our approach and describe precomputation optimizations. Section 9 presents simulation results. Section 10 describes a use case implementation of our system, focused on parking tolls for high-occupancy vehicles. Finally, Section 11 summarizes conclusions. The Appendix contains the proof of the IBDT security theorem and discusses details on how to combine the IBDT primitive with our key management proposal.

The protocol suite introduced here is a generalization of a specific protocol for high-occupancy vehicle toll discounts for which we filed patent [20]. A previous and partial conference version of this paper was presented in [19]. Sections 2, 3.1, 4, 9, 10 and the security proof in the Appendix are entirely new to this journal version. Furthermore, we now use asymmetric pairings (instead of the symmetric ones in the preliminary version), which have a more efficient arithmetic for the same security level [14], especially since the recent attacks to the discrete logarithm problem [25].

2 Related Work

2.1 Group size accreditation systems

In order to offer group discounts, a system to count group members is needed. This 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 [12]. A mechanism that is highly effective against cheaters is described in [16], 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.

2.2 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 4.2 for a description of it and Section 5 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 which 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 [23] in bilinear groups and improvements thereof which specifically try to improve GS proofs for this type of "threshold statement" [34,21]. 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 ([26], and improvements thereof [27,29]) allow only proving membership in linear spaces of a discrete-logarithm group $\mathbb{G}$ or are designated-verifier [1], 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 [15], 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" [3,28], 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

[1] Even if there are ring signatures of constant size [18], we are not aware of constant-size identity-based threshold ring signatures.

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 Background

3.1 Preliminaries

We use asymmetric bilinear pairings, which are mappings $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are multiplicative groups[2] of prime order p, e is efficiently computable and it holds that $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for any $(g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$, $a, b \in \mathbb{Z}$ and $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ whenever $g_1 \neq 1_{\mathbb{G}_1}$ and $g_2 \neq 1_{\mathbb{G}_2}$.

We will treat vectors as column vectors. For any $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top \in \mathbb{Z}_p^n$, and any element g of a group $\mathbb{G}_i$, $g^{\boldsymbol{\alpha}}$ stands for $(g^{\alpha_1}, \ldots, g^{\alpha_n})^\top \in \mathbb{G}_i^n$. The inner product of $\mathbf{a}, \mathbf{z} \in \mathbb{Z}_p^n$ is denoted as $\langle \mathbf{a}, \mathbf{z} \rangle = \mathbf{a}^\top \mathbf{z}$. Given $g^{\mathbf{a}}$ and $\mathbf{z}$, $(g^{\mathbf{a}})^{\mathbf{z}} := g^{\langle \mathbf{a}, \mathbf{z} \rangle}$ is computable without knowing $\mathbf{a}$. For equal-dimension vectors $\mathbf{A}$ and $\mathbf{B}$ of exponents or group elements, $\mathbf{A} \cdot \mathbf{B}$ stands for their component-wise product. Given a set $S \subset \mathbb{Z}_p$, and some $i \in S$, the i-th Lagrange basis polynomial is $\Delta_i^S(X) = \prod_{j \in S \setminus \{i\}} (X - j)/(i - j)$.

We will use the following complexity assumption on the hardness of the asymmetric n-Diffie-Hellman Exponent problem, which is an adaptation of the n-Diffie-Hellman Exponent problem in [10] to asymmetric bilinear pairings and which can be proven generically hard following a result of [8].

Definition 1 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 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] When discussing asymmetric bilinear pairings, the term group is used in its algebraic acceptation. Otherwise, in this paper we use group in its ordinary acceptation of a number of persons or entities.

3.2 Signature schemes

The proposed group size accreditation method relies on a new cryptographic primitive called identity-based dynamic threshold (*IBDT*) signature scheme. This primitive combines the properties of threshold signatures and identity-based cryptography, which are reviewed next.

Threshold cryptography, specifically a threshold signature scheme, typically relies on t-out-of-n secret sharing schemes, such as the ones introduced in [5] and [36] by Blakley and Shamir, respectively. What defines a t-out-of-n threshold signature scheme is the requirement that t secret keys (and hence t different participants), out of n distributed secret keys, are used to produce valid signatures, which are then verified using a unique *master* public key. The *IBDT* primitive additionally has the property that the threshold t is dynamic and chosen (and declared) when each signature is computed, instead of being a public parameter chosen at set-up time. It is this feature, *i.e.* choosing and declaring the number of signers to produce a valid signature, what allows us to certify the number of members of a group.

Identity-based cryptography allows the use of arbitrary strings, typically related to the identity **id** of an entity, such as the name or the email address of some user, as the public keys in asymmetric cryptographic schemes. Identity-based cryptography was theorized by Shamir in [37] in 1985 as a response to the increasing complexity of PKIs, but the first instantiation, based on the Weil pairing, was developed by Boneh *et al.* in [9] in 2001. A certification authority (CA) uses then the identity **id** of the user as her public key $ik^{\mathbf{id}}$, to compute her associated secret key $sk^{\mathbf{id}}$. Identity-based public key signature schemes offer a great flexibility in key generation and management. We leverage this feature in our group accreditation method and, particularly, in our key distribution mechanism.

3.3 Anonymous payment mechanisms

The method we present in this paper allows a group to prove its size anonymously. When this is done to obtain a group discount, group members must thereafter pay a fee that depends on the group size. After preserving member anonymity in the proof of the size, it would be pointless to pay the fee using a non-anonymous payment protocol (such as credit card, PayPal, etc.).

Therefore, we rely on anonymous payment mechanisms together with our group accreditation method to provide anonymity to group discounts. While these mechanisms are not the focus of the present work, we provide a short list of candidate methods, electronic or

not, that can fulfill our requirements. The simplest approach for an anonymous payment method is to use cash. While this is the most straightforward strategy, it is not suitable for on-line transactions. Electronic cash protocols, such as [13] and newer and more sophisticated approaches are a good candidate for this role. However, to the best of our knowledge, there is no fully deployed e-cash mechanism of this sort which we could resort to. Nowadays, Bitcoin [32] is a well-established electronic currency and, although it is not anonymous by design [35], it can be a good solution if accompanied by careful key management policies. Also, extensions of the original protocol such as Zerocoin [31] or the more recent and significantly faster Zerocash [4], provide anonymity by design.

For completeness, we propose in this work to adopt 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 $\mathsf{Pay.Code}$ 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 [33].

3.4 Communication technologies

Communication among the members (their devices) of a group is required to participate in our accreditation mechanism. 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, [6]) 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).

4 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 setup* 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 4.4 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 $\mathsf{IBDT.Setup}$, 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 $SK_{\mathbf{id}} \leftarrow \mathsf{IBDT.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \mathbf{id})$.

4.1 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 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 2 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 $(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBDT.Setup}(1^\lambda, \mathcal{ID}, n)$ 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 $SK_{\mathbf{id}} \leftarrow \mathsf{IBDT.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \mathbf{id})$.
- ***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:

- $\mathsf{IBDT.Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}^\star, \sigma^\star, \Gamma^\star) = 1$, 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

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

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

4.2 An instance of an IBDT signature scheme

We instantiate an IBDT signature scheme by adapting the threshold attribute-based signature scheme of [24], which builds on the attribute-based encryption scheme of [2]. 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 [24] 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 Appendix B, 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 [24] 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 [38] 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 [7]. 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 $\alpha, \alpha_0 \leftarrow \mathbb{Z}_p$, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^\top \leftarrow \mathbb{Z}_p^N$, where $N = n + 1$. It then computes $e(g_1, g_2)^\alpha$, $h_0 = g_1^{\alpha_0}$, $f_0 = g_2^{\alpha_0}$, $\mathbf{H} = (h_1, \ldots, h_N)^\top = g_1^{\boldsymbol{\alpha}}$ and $\mathbf{F} = (f_1, \ldots, f_N)^\top = g_2^{\boldsymbol{\alpha}}$.
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 $\mathbf{U} := (g_1^{\tilde{u}_0}, g_1^{\tilde{u}_1})$ and $\mathbf{V} := (g_2^{\tilde{u}_0}, g_2^{\tilde{u}_1})$.
The master secret key is defined to be $\mathsf{msk} = (g_1^\alpha, Q)$ and the master public key is

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

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

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

► $\mathsf{Sign}(\mathsf{pms}, \mathsf{mpk}, SK_{\mathbf{id}}, \mathsf{Msg}, \Gamma, \mathbf{id})$: 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 $s = |S| \le n$, the algorithm first computes $M = H(\mathsf{Msg}, \Gamma)$. It defines $\mathbf{Y} = (y_1, \ldots, y_N)^\top$ as the vector containing the coefficients of the polynomial

$$P_S(Z) = \sum_{i=1}^{N} y_i Z^{i-1} = \prod_{\mathbf{id} \in S} (Z - \mathbf{id}) \prod_{d \in \mathcal{D}_{n-s}} (Z - d). \quad (2)$$

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

$$D'_{\mathbf{id},1} = D_{\mathbf{id},1} \cdot \prod_{i=1}^{N-1} K_{\mathbf{id},i}^{y_{i+1}} = g_1^{Q(\mathbf{id})} \cdot \big(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\big)^{r_{\mathbf{id}}}. \quad (3)$$

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

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

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

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

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

$$\sigma_2 = \prod_{\mathbf{id}\in S} \left(\sigma_{\mathbf{id},2}\right)^{\Delta_{\mathbf{id}}^{S\cup\mathcal{D}_{n-s}}(0)} \prod_{j=1}^{n-s} \left(D_{\mathbf{id}',j,2}\right)^{\Delta_{d_j}^{S\cup\mathcal{D}_{n-s}}(0)} = g_1^r,$$

$$\sigma_3 = \prod_{\mathbf{id}\in S_t} \left(\sigma_{\mathbf{id},3}\right)^{\Delta_{\mathbf{id}}^{S\cup\mathcal{D}_{n-s}}(0)} = g_1^z,$$

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 $\mathbf{Y} = (y_1, \ldots, y_N)^\top$ from the polynomial $P_S(Z)$ as per Equation (2). The algorithm accepts the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ as valid and thus outputs 1 if and only if

$$e(g_1, g_2)^{\alpha} \stackrel{?}{=} \frac{e(\sigma_1, g_2)}{e\left(\sigma_2, f_0 \cdot \prod_{i=1}^{N} f_i^{y_i}\right) \cdot e\left(\sigma_3, (v_0^M \cdot v_1)\right)}. \quad (4)$$

4.3 Security of the IBDT instance

In the Appendix, we give the proof of the following theorem regarding the unforgeability of the IBDT signature scheme in Section 4.2.

Theorem 1 *The IBDT signature scheme of Section 4.2 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.*

4.4 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 [30]).
- 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 5 below, η is 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 5), 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 5 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 in Appendix B.

5 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 Appendix B.

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 the 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.

6 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 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* **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* IBDT.Setup*. SP makes* pms *and* mpk *available.*
3. *SP generates and publishes the parameters of a public-key encryption scheme* Π'.

Protocol 2 ***Registration protocol.***

1. *A user with identifier* **id** *authenticates to the service provider, face-to-face or otherwise.* **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* **id***, in the way explained in Section 5.*
3. *SP computes the secret keys associated to* $\mathbf{IK_{id}}$ *as per Algorithm* IBDT.Keygen*:*

$$\mathbf{SK_{id}} = \left(\mathsf{sk}_1^{d_1^{\mathbf{id}}}, \ldots, \mathsf{sk}_\ell^{d_\ell^{\mathbf{id}}}\right).$$

4. **id** *downloads the smartphone app* $\mathsf{App}_{\mathbf{id}}$ *and, using* $\mathsf{pin}_{\mathbf{id}}$*, 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* 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* $\mathbf{IK}_{\mathbf{id}_1}, \ldots, \mathbf{IK}_{\mathbf{id}_t}$ *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* 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* 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.*

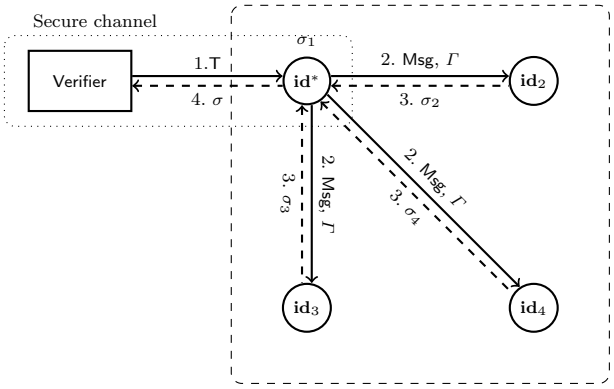

Fig. 1 Group size accreditation protocol

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.*

7 Security and Privacy

Our mechanism offers security and privacy by design:

- As stated by Theorem 1 and proven in the Appendix, 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 5. 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.

Note 1 (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.

Note 2 (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.

8 Performance

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 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, $t \leq n$.

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 (2) and (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 2, where SignPrecomputation and CombPrecomputation are the respective precomputation phases, and FastSign and FastComb are the respective interactive phases run during Protocol 5.

9 Simulations

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 [17]. Our 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 [30].

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.

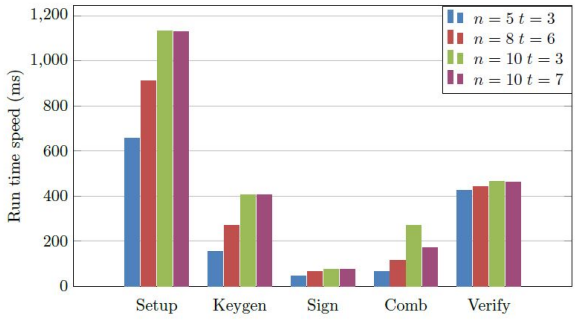

Fig. 2 Execution times of the protocol without precomputation for different values of n and t

Execution times of the algorithms, without precomputation, are shown in Figure 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 shows the execution times of the Sign and Comb algorithms when precomputation is performed. Thus, these algorithms are shown split as (SignPrecomputation, FastSign) and (CombPrecomputation, FastComb).

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 offline and/or as a background process when the group is formed.

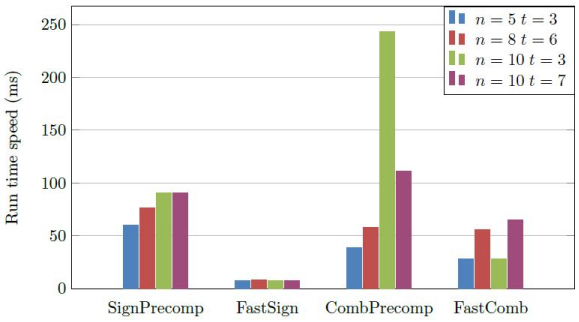

Fig. 3 Execution times of the protocol with precomputation for different values of n and t

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 8 and Table 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 4.

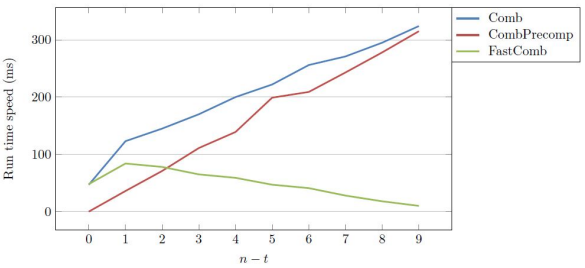

Fig. 4 Comb times as a function of $(n - t)$

10 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 6;
- 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 10 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.

11 Conclusions and Future Work

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. 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, 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.

Future work will involve integrating the proposed system 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.

Acknowledgments and disclaimer

The following funding sources are acknowledged: Google (Faculty Research Award to the first author), Government of Catalonia (ICREA Acadèmia Prize to the first author and grant 2014 SGR 537), Spanish Government (projects TIN2014-57364-C2-1-R "SmartGlacis" and TIN2015-70054-REDC), and European Commission (projects H2020-644024 "CLARUS" and H2020-700540 "CANVAS"). The authors are with the UNESCO Chair in Data Privacy. The views in this paper are the authors' own and do not necessarily reflect the views of UNESCO or any of the funders.

References

1. M. Abdalla, F. Benhamouda and D. Pointcheval, "Disjunctions for Hash Proof Systems: New Constructions and Applications," in *Advances in Cryptology - Eurocrypt'15*, Part II LNCS 9057, pp. 69-100, (2015).
2. N. Attrapadung, B. Libert and E. de Panafieu. "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *Public Key Cryptography—PKC'11*, LNCS 6571, pp. 90–108, 2011.
3. M. Bellare, H. Shi, and C. Zang, "Foundations of group signatures: the case of dynamic groups," in *CT - RSA '05*, LNCS 3376, pp. 136–153, Springer, 2005.
4. E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*, pp. 459–474, IEEE, 2014.
5. G. R. Blakley, "Safeguarding cryptographic keys," in *Proceedings of the National Computer Conference*, pp. 313–317, New York: AFIPS Press, 1979.
6. Bluetooth SIG, *Specification of the Bluetooth System*, 2013. Available in `https://www.bluetooth.org/en-us/specification/adopted-specifications`.
7. D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *Advances in Cryptology–Eurocrypt'04*, LNCS 3027, pp. 223–238, Springer, 2004.
8. D. Boneh, X. Boyen and E.-J. Goh, "Hierarchical identity-based encryption with constant size ciphertext," in *Advances in Cryptology–Eurocrypt'05*, LNCS 3494, pp. 440–456, Springer, 2005.
9. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology–Crypto'01*, LNCS 2139, pp. 213–229, Springer, 2001.
10. D. Boneh, C. Gentry and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Advances in Cryptology-Crypto'05*, LNCS 3621, pp. 258–275, Springer, 2005.
11. D. Boneh and M. Hamburg, "Generalized identity-based and broadcast encryption schemes," in *Advances in Cryptology- Asiacrypt'08*, LNCS 5350, pp. 455–470, Springer, 2008.
12. 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 April 12, 2018.

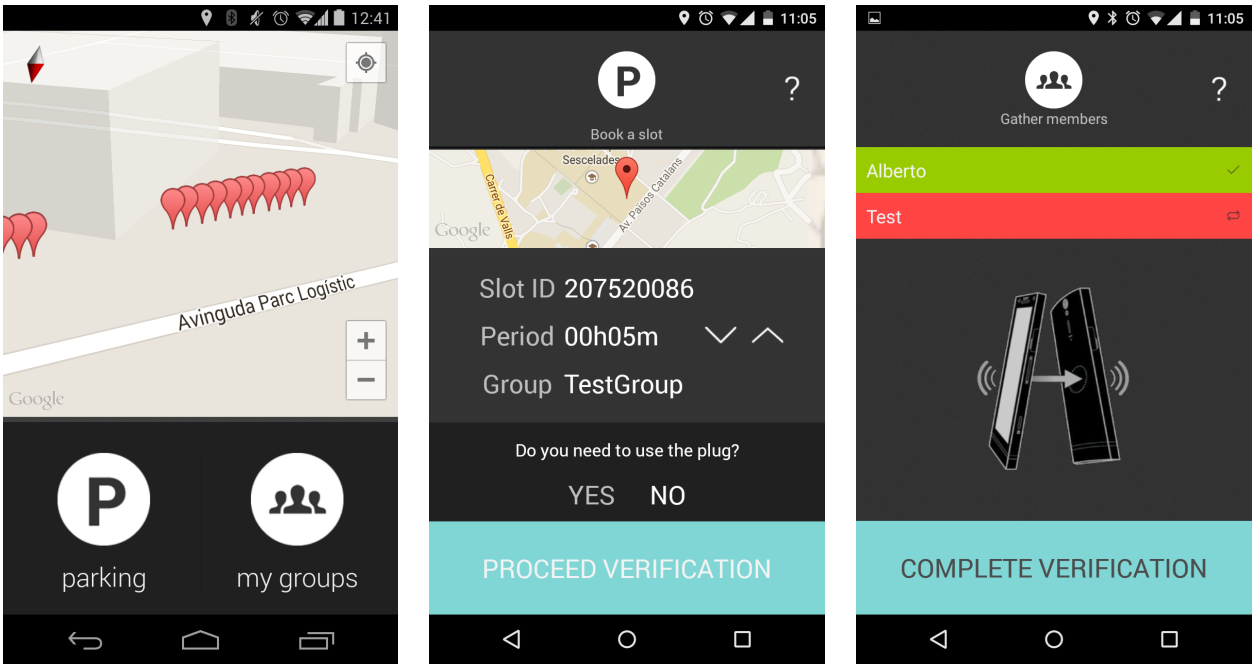

Fig. 5 HOV pilot passenger application. Left, locating free HOV spaces. Center, booking a space and starting group size accreditation. Right, completing accreditation of a group of two passengers.

13. D. Chaum, A. Fiat and M. Naor, "Untraceable electronic cash," in *Advances in Cryptology–Crypto'88*, LNCS 403, pp. 319–327, Springer, 1990.
14. L. Chen, P. Morrissey and N. P. Smart, "Pairings in trusted computing", in *Pairing-Based Cryptography – Pairing 2008*, LNCS 5209, pp. 1–17, Springer, 2008.
15. R. Cramer, I. Damgård and B. Schoenmakers. "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols", in *Advances in Cryptology–Crypto'94*, LNCS 839, pp. 174–187, Springer, 1994.
16. 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.
17. A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *2011 Symposium on Computers and Communication (ISCC)*, pp. 850–855, IEEE, 2011. Available in `http://gas.dia.unisa.it/projects/jpbc/`.
18. Y. Dodis, A. Kiayias, A. Nicolosi and V. Shoup, "Anonymous identification in ad-hoc Groups," in *Advances in Cryptology–Eurocrypt'04*, LNCS 3029, pp. 609–627, Springer, 2004.
19. 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.
20. 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.
21. A. González, A.Hevia and C. Ràfols, "QA-NIZK Arguments in Asymmetric Groups: New Tools and New Constructions", in *ASIACRYPT 2015*, Part I LNCS 9452, pp. 605-629, Springer, 2015.
22. V. Goyal, O. Pandey, A. Sahai and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *ACM CCS'06*, pp. 89–98, ACM Press, 2006.
23. J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Advances in Cryptology–Eurocrypt'08*, LNCS 4965, pp. 415–432, Springer, 2008.
24. 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.
25. A. Joux, "A new index calculus algorithm with complexity L(1/4+o(1)) in small characteristic," in *Selected Areas in Cryptography - SAC 2013*, LNCS 8282, pp. 355–379, Springer, 2014.
26. C.S. Jutla and A. Roy, "Shorter quasi-adaptive NIZK proofs for linear subspaces", in *ASIACRYPT 2013*, LNCS 8269, pp. 1-20, Springer, 2013.
27. C.S. Jutla and A. Roy, "Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces", in *Advances in Cryptology–Crypto'14*, LNCS 8617, pp. 295-312, Springer, 2014.
28. B. Libert, S. Ling, F. Mouhartem, K. Nguyen and H. Wang, "Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions," IACR Cryptology ePrint Archive, 2016.
29. 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'14*, LNCS 8441, pp. 514-532, Springer, 2014.
30. B. Lynn, *On the Implementation of Pairing-based Cryptosystems*, Doctoral dissertation, Stanford University, 2007.
31. I. Miers, C. Garman, M. Green and A. D. Rubin, "Zerocoin: anonymous distributed e-cash from bitcoin," in *2013*

Symposium on Security and Privacy, pp. 397–411, IEEE, 2013.
32. S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Consulted*, vol. 1, 2008. Available in `http://www.bitcoin.org/bitcoin.pdf`.
33. Paysafecard, `http://paysafecard.com`, checked Dec. 1, 2014.
34. C. Ràfols, "Stretching Groth-Sahai proofs: NIZK proofs of partial satisfiability," in *TCC'15*, LNCS 9015, pp. 247–276, Springer, 2015.
35. 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.
36. A. Shamir, "How to share a secret," *Communications of the ACM*, 22:612–613, 1979.
37. A. Shamir, "Identity based cryptosystems and signature schemes," in *Advances in Cryptology–CRYPTO 1984*, LNCS 196, pp. 47–53, Springer, 1985.
38. B. Waters. "Efficient identity-based encryption without random oracles," in *Eurocrypt'05*, LNCS 3494, pp. 114–127, Springer, 2005.

A Proof of Theorem 1

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 $\boldsymbol{\gamma}$ the vector $\boldsymbol{\gamma} := (\gamma, \gamma^2, \ldots, \gamma^N)$ 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 $\mathbf{id}$.

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 $\mathbf{Y}$ associated with the polynomial $P_{S^\star}(Z)$ according to Expression (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 $\boldsymbol{\theta} \leftarrow \mathbb{Z}_p^N$ and computes $\mathbf{H} = (h_1, \ldots, h_N)^\top = g_1^{\boldsymbol{\gamma}} \cdot g_1^{\boldsymbol{\theta}}$, $\mathbf{F} = (f_1, \ldots, f_N)^\top = g_2^{\boldsymbol{\gamma}} \cdot g_2^{\boldsymbol{\theta}}$ (which implicitly sets $\boldsymbol{\alpha} = \boldsymbol{\gamma} + \boldsymbol{\theta}$), $h_0 = g_1^{\theta_0} \cdot g_1^{-\langle \boldsymbol{\gamma}, \mathbf{Y} \rangle}$, $f_0 = g_2^{\theta_0} \cdot g_2^{-\langle \boldsymbol{\gamma}, \mathbf{Y} \rangle}$ 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 $\mathsf{mpk} = \big(e(g_1, g_2)^\alpha, h_0, f_0, \mathbf{H}, \mathbf{F}, \mathbf{U}, \mathbf{V}, \mathcal{D}, H\big)$ is given to $\mathcal{F}$.

In the following, for any $\omega \in \mathbb{Z}_p$, we define the vector $\mathbf{X}_\omega^n = (1, \omega, \ldots, \omega^{n-1})^\top$. We note that, given any set $S \subset \mathbb{Z}_p$ of cardinality less than n, the vectors $\{\mathbf{X}_\omega^n\}_{\omega \in S}$ 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 $\mathbf{X}_0^n = (1, 0, \ldots, 0)^\top$ cannot be in the span of the vectors $\{\mathbf{X}_\omega^n\}_{\omega \in (S^\star \cap \Omega) \cup \mathcal{D}_{n-s^\star}}$. Pick $\mu \leftarrow \mathbb{Z}_p^*$. We conclude that there exists an efficiently computable vector $\boldsymbol{\tau}$ which is uniform conditioned on $\langle \mathbf{X}_\omega^n, \boldsymbol{\tau} \rangle = 0$ for any $\omega \in (S^\star \cap \Omega) \cup \mathcal{D}_{n-s^\star}$ and $\langle \mathbf{X}_0^n, \boldsymbol{\tau} \rangle = \mu \neq 0$ (according to Proposition 1 in [22]).

To construct a secret key, $\mathcal{B}$ has to define a random vector $\mathbf{u}$ which satisfies the constraint $\langle \mathbf{X}_0^n, \mathbf{u} \rangle = \alpha$, *i.e.* $\mathbf{u} = (\alpha, \beta_1, \ldots, \beta_{n-1})^\top$. This vector defines the coefficients of $Q[X]$. To this end, $\mathcal{B}$ proceeds as in the proof of Theorem 3 in [22], by implicitly setting $\mathbf{u}$ as $\mathbf{u} = \mathbf{v} + \psi \cdot \boldsymbol{\tau}$, where $\mathbf{v} = (v_1, \ldots, v_n)^\top \in \mathbb{Z}_p^n$ is a randomly chosen vector and $\psi = (\alpha - v_1)/\mu$, so that $\langle \mathbf{X}_0^n, \mathbf{u} \rangle = \alpha$. The task of $\mathcal{B}$ is thus to compute (without knowing the vector $\mathbf{u}$) 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

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

where $Q(\omega) = \langle \mathbf{X}_\omega^n, \mathbf{u} \rangle$, 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 $Q(\mathbf{id}) = \langle \mathbf{X}_{\mathbf{id}}^n, \mathbf{u} \rangle = \langle \mathbf{X}_{\mathbf{id}}^n, \mathbf{v} \rangle$ 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 [11]. 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 $\boldsymbol{\xi} = \xi_1 \cdot (1, \mathbf{id}, \ldots, \mathbf{id}^{N-1})^\top$, which satisfies $\boldsymbol{\xi}^\top M_{\mathbf{id}} = \mathbf{0}$ while $\langle \mathbf{Y}, \boldsymbol{\xi} \rangle = \xi_1 \cdot P_{S^\star}(\mathbf{id}) \neq 0$. The simulator $\mathcal{B}$ computes

$$\big(D_{\mathbf{id},1}^*, D_{\mathbf{id},2}^*\big) = \Big(g_1^{\alpha} \cdot h_0^{\tilde{r}_{\mathbf{id}}},\ g_1^{\tilde{r}_{\mathbf{id}}}\Big)$$
$$\text{and} \qquad \big(K_{\mathbf{id},1}^*, \ldots, K_{\mathbf{id},N-1}^*\big)^\top = g_1^{\tilde{r}_{\mathbf{id}} M_{\mathbf{id}}^\top \boldsymbol{\alpha}}, \qquad (5)$$

with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^\top$ and where the exponent $\tilde{r}_{\mathbf{id}}$ is defined as $\tilde{r}_{\mathbf{id}} = r + \delta_0 \langle (\gamma^N, \gamma^{N-1}, \ldots, \gamma)^\top, \boldsymbol{\xi} \rangle / \langle \mathbf{Y}, \boldsymbol{\xi} \rangle$ for some $r \leftarrow \mathbb{Z}_p$ chosen by $\mathcal{B}$. Since $g^{M_{\mathbf{id}}^\top \boldsymbol{\alpha}} = (h_1^{-\mathbf{id}} h_2, \ldots, h_1^{-\mathbf{id}^{N-1}} h_N)^\top$, if we can argue that both expressions in (5) are computable by $\mathcal{B}$, we will have concluded the first step.

For any $\mathbf{x} \in \mathbb{Z}_p^N$, the coefficient of γ^{N+1} in the product $\tilde{r}_{\mathbf{id}} \langle \mathbf{x}, \boldsymbol{\gamma} \rangle$ is $\delta_0 \langle \mathbf{x}, \boldsymbol{\xi} \rangle / \langle \mathbf{Y}, \boldsymbol{\xi} \rangle$. The reason why $\mathcal{B}$ can compute the second factor of $D_{\mathbf{id},1}^*$ in (5) is that the coefficient of $g^{\gamma^{N+1}}$ in $D_{\mathbf{id},1}^*$ is 0. Indeed, $D_{\mathbf{id},1}^* = g^{\alpha} \cdot h_0^{\tilde{r}_{\mathbf{id}}} = z_{N+1}^{\delta_0} \cdot$

$\left(g^{\theta_0} \cdot g^{-\langle \boldsymbol{\gamma}, \mathbf{Y} \rangle}\right)^{\tilde{r}_{\mathbf{id}}}$ and the coefficient of γ^{N+1} is $-\delta_0$ in the product $-\tilde{r}_{\mathbf{id}}\langle \boldsymbol{\gamma}, \mathbf{Y} \rangle$, as we can see by applying the observation above in the case $\mathbf{x} = \mathbf{Y}$. Since $M_{\mathbf{id}}^\top \boldsymbol{\xi} = \mathbf{0}$, by applying the above observation to the case where $\mathbf{f}^\top$ 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 $g_1^{\tilde{r}_j \cdot M_{\mathbf{id}}^\top \boldsymbol{\alpha}}$, which is computable. This concludes the first step of the key generation process. In the second step, we just have to turn $\left(D^*_{\mathbf{id},1}, D^*_{\mathbf{id},2}, \{K^*_{\mathbf{id},i}\}_{i=1}^{N-1}\right)$ into a suitable key component. Note that

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

where $\kappa_1 = (\sum_{j=1}^n \mathbf{id}^{j-1}\tau_j)\cdot\mu^{-1}$ and $\kappa_2 = \mu^{-1}\cdot\sum_{j=1}^n \mathbf{id}^{j-1}$ $\left(\mu v_j - v_1\tau_j\right)$ 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

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

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 $\mathbf{Y} = (y_1, \ldots, y_N)^\top$ whose coordinates are the coefficients of the polynomial $P_S(Z)$ which is obtained following Expression (2), by augmenting S with $n-s$ dummy signers. Recall that $\mathcal{B}$ has to generate a signature of the form

$$\sigma_1 = g_1^\alpha \cdot \big(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\big)^r \cdot (u_0^M \cdot u_1)^{\tilde{z}}, \quad \sigma_2 = g_1^r, \quad \sigma_3 = g_1^{\tilde{z}}, \tag{6}$$

for some $r, \tilde{z} \leftarrow \mathbb{Z}_p$. To this end, $\mathcal{B}$ uses the usual technique (which dates back to [7]) 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 (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 $\mathbf{Y} = (y_1, \ldots, y_N)^\top$ derived from $\Gamma^\star$ is such that $f_0 \cdot \prod_{i=1}^N f_i^{y_i} = g_2^{\theta_0 + \langle \boldsymbol{\theta}, \mathbf{Y} \rangle}$, $v_0^M \cdot v_1 = v_0^{M^\dagger} \cdot v_1 = g_2^{t_1}$, and $\sigma^\star$ satisfies Equation (4), we must have

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

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

B IBDT with Special Key Management

As mentioned in Section 4.4, our key management requires slightly modifying the IBDT scheme to use it for $\ell > 1$ (for $\ell = 1$ the scheme would be the same). Indeed, according to our key management structure, a user $\mathbf{id}_i$ receives all the secret keys associated to a vector of identity-based public keys $\{\mathsf{ik}_1^{\mathbf{id}_i}, \ldots, \mathsf{ik}_\ell^{\mathbf{id}_i}\}$ and we want to make sure that a valid signature is constructed from the secret keys corresponding to s *different* users $\mathbf{id}_1, \ldots, \mathbf{id}_s$ (and not, for example, by combining s different secret keys of a single user).

It is straightforward to modify the IBDT scheme for this special key management. Given a user $\mathbf{id}$ 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 described in Section 4, 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^λ, $\mathcal{ID}$, n,ℓ): 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 secret key vector $\mathbf{SK_{id}} := (\{D_{\mathbf{id},1,k}\}_{k=1}^\ell, D_{\mathbf{id},2}, K_{\mathbf{id},1}, \ldots, K_{\mathbf{id},N-1}, \{\{D_{\mathbf{id},j,1,k}\}_{k=1}^\ell, D_{\mathbf{id},j,2}, K_{\mathbf{id},j,1}, \ldots,$

$K_{\mathbf{id},j,N-1}\}_{j=1\ldots n-1}$) by picking fresh random elements $r_{\mathbf{id}}, r_{\mathbf{id},1}, \ldots, r_{\mathbf{id},n-1} \leftarrow \mathbb{Z}_p$ and setting:

$$\begin{aligned}
&\{D_{\mathbf{id},1,k} = g_1^{Q_k(\mathbf{id})} \cdot h_0^{r_{\mathbf{id}}}\}_{k=1}^{\ell},\\
&D_{\mathbf{id},2} = g_1^{r_{\mathbf{id}}},\\
&\left\{K_{\mathbf{id},i} = \left(h_1^{-\mathbf{id}^i} \cdot h_{i+1}\right)^{r_{\mathbf{id}}}\right\}_{i=1,\ldots,N-1},\\
&\left\{\{D_{\mathbf{id},j,1,k} = g_1^{Q_k(d_j)} \cdot h_0^{r_{\mathbf{id},j}}\}_{k=1}^{\ell}, D_{\mathbf{id},j,2} = g_1^{r_{\mathbf{id},j}},\right.\\
&\left.\left\{K_{\mathbf{id},j,i} = \left(h_1^{-d_j^i} \cdot h_{i+1}\right)^{r_{\mathbf{id},j}}\right\}_{i=1,\ldots,N-1}\right\}_{j=1,\ldots,n-1}. \quad (7)
\end{aligned}$$

Algorithm Sign is the same as specified in Section 4 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.

Table 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

Table 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