Privacy-aware Loyalty Programs

Alberto Blanco-Justicia and Josep Domingo-Ferrer

Universitat Rovira i Virgili, UNESCO Chair in Data Privacy, Department of Computer Engineering and Mathematics, Av. Països Catalans 26, E-43007 Tarragona, Catalonia
E-mail {alberto.blanco,josep.domingo}@urv.cat

Abstract

Vendors use loyalty programs as a mechanism to incentivize returning customers, whose repeated transactions provide sustained profit and information on the customers' preferences. Such programs have become widespread, but they are facing criticism by business experts and consumer associations: since they facilitate profiling, a loss of consumer privacy ensues. We propose a protocol for privacy-aware loyalty programs that allows vendors and consumers to enjoy the benefits of loyalty (returning customers for the vendor and rewards for the consumers, respectively), while allowing consumers to stay anonymous and empowering them to decide how much of their profile they reveal to the vendor. The vendor must offer additional reward if he wants to learn more details on the consumer's profile. Our protocol is based on partially blind signatures and generalization of product receipts, and provides anonymity to consumers and their purchases, while still allowing negotiated consumer profiling. We provide empirical results that confirm the viability of our approach.

Key words: Loyalty programs, customer privacy, anonymization, blind signatures, smartphones.

1 Introduction

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

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

Although loyalty programs have become widespread, they are experiencing a loss of active participants and they have been criticized by business experts and consumer associations. Criticism is mainly due to privacy issues, because it is not always clear whether the benefits vendors offer in their loyalty programs are worth the loss of consumer privacy caused by profiling [27,34,1,17].

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

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

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

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

- To prevent straightforward profiling by the vendor, payment should be anonymous. In online stores, to completely achieve anonymity, the buyers should use some kind of anonymous payment system, such as Bitcoin [25], Zerocoin [24], some other form of electronic cash [14], or simply scratch cards with prepaid credit anonymously bought, say, at a newsstand. In physical stores, it would be enough to pay with cash.
- Consumers should not be able to leverage their anonymity to reveal forged profiles to the vendor, which would earn them rewards without actually revealing anything on their real purchase pattern.

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

The paper is organized as follows. Section 2 starts recalling the functionalities of a conventional loyalty program; then it formalizes the notion of privacy-aware loyalty program; after that, it goes on to present the security and functionality requirements that our new privacy-aware protocol suite should satisfy. Section 3 reviews related work. Section 4 describes the cryptographic background used in our proposed construction, including bilinear maps, partially blind signatures, and zero-knowledge proofs. In Section 5, we explain how we use generalization of purchase receipts while preventing the existence of several generalizations of each purchase receipt from being abused to submit the receipt more than once to get loyalty points. In Section 6 we introduce anonymous tokens with controlled linkability based on partially blind signatures. In Section 7 we present our privacy-aware loyalty program protocol suite, that builds on the tools described in Sections 4, 5 and 6. In Section 8 we analyze the computational complexity and the security of the suite. In Section 9 we present experimental results, including an implementation on smartphones and an analysis of deployability in physical and online stores. Section 10 presents an extension of our construction that guarantees untransferability of purchase receipts and/or loyalty points. Finally, Section 11 summarizes conclusions.

A previous and partial version of this paper was presented in the conference paper [7]. Sections 3, 9 and Section 10 are entirely new to this journal version; also, substantial additions have been made to Section 2 (to which formalization and the untransferability requirement have been added), Section 4.2, Section 8 and Section 11. Furthermore, whereas [7] relied on symmetric pairings as the underlying cryptographic building block, we use here asymmetric pairings for increased security [15].

2 Loyalty programs

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

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

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

2.1 A privacy-aware alternative

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

Definition 1. *(Privacy-aware Loyalty Program) A Privacy-aware Loyalty Program scheme has three participants: a key dealer or certification authority $\mathcal{CA}$, a vendor $\mathcal{V}$, and a customer $\mathcal{C}$. $\mathcal{V}$ keeps a set $\mathcal{DB}$ of submitted tokens that is initially empty. The scheme consists of seven protocols (*Setup, VendorSetup, Enroll, Buy, Submit, Issue, Redeem*):*

- Setup *is a probabilistic polynomial-time algorithm run by $\mathcal{CA}$ in which, on input a security parameter γ, outputs (and publishes) the system parameters* params.
- VendorSetup *is a probabilistic polynomial-time key generation algorithm. $\mathcal{V}$ is certified as a legitimate vendor and obtains* params *and a key pair $(pk_{\mathcal{V}}, sk_{\mathcal{V}})$ from $\mathcal{CA}$.*
- Enroll *is a protocol run by $\mathcal{C}$ whereby $\mathcal{C}$ obtains access to the loyalty program, typically by registering and obtaining* params *and $pk_{\mathcal{V}}$.*
- Buy *is a probabilistic polynomial-time interactive protocol run between $\mathcal{V}$ and $\mathcal{C}$. The public inputs of both $\mathcal{C}$ and $\mathcal{V}$ contain a product name p. The private input of $\mathcal{C}$ contains a message y, and that of $\mathcal{V}$ contains the private key $sk_{\mathcal{V}}$. When the protocol finishes, the private output of $\mathcal{C}$ contains either* `fail` *or a list of receipt tokens $(R_i)_{i=1,\ldots,n}$.*
- Submit *is a polynomial-time algorithm that takes $pk_{\mathcal{V}}$, a receipt token R and $\mathcal{DB}$ as inputs.* Submit *outputs either* `accept` *if the signature on the token R is valid and $R \notin \mathcal{DB}$ or* `reject` *otherwise. An accepted submitted token is thereafter invalidated by adding it to $\mathcal{DB}$.*
- Issue *is a probabilistic polynomial-time interactive protocol run between $\mathcal{V}$ and $\mathcal{C}$. The public inputs of both $\mathcal{C}$ and $\mathcal{V}$ contain a value c of loyalty points. The private input of $\mathcal{C}$ contains a message y, and that of $\mathcal{V}$ contains a private key $sk_{\mathcal{V}}$. When the protocol finishes, the private output of $\mathcal{C}$ contains either* `fail` *or a loyalty points token P.*
- Redeem *is a polynomial-time algorithm that takes $pk_{\mathcal{V}}$ and loyalty points token P and $\mathcal{DB}$ as inputs.* Redeem *outputs either* `accept` *if the signature on the token P is valid and $P \notin \mathcal{DB}$ or* `reject` *otherwise. A redeemed token is then invalidated by adding it to $\mathcal{DB}$.*

Definition 2. *(Correctness and security of a Privacy-aware Loyalty Program) A Privacy-aware Loyalty Program is considered correct and secure if, given* $\mathsf{params} \leftarrow \mathsf{Setup}(\gamma)$*,* $(pk_{\mathcal{V}}, sk_{\mathcal{V}}) \leftarrow \mathsf{VendorSetup}(\mathsf{params})$*, it fulfills the following properties:*

- *(Correctness) For any product name* p*, any private input* y *of* $\mathcal{C}$*, a private key* $sk_{\mathcal{V}}$*, and a receipt token* $R \leftarrow \mathsf{Buy}(p, y, sk_{\mathcal{V}})$ *such that* $R \notin \mathcal{DB}$*, the execution of* $\mathsf{Submit}(pk_{\mathcal{V}}, R, \mathcal{DB})$ *must return* `accept` *with overwhelming probability. Likewise, for any value* c*, any private input* y *of* $\mathcal{C}$*, a private key* $sk_{\mathcal{V}}$*, and a loyalty points token* $P \leftarrow Issue(c, y, sk_{\mathcal{V}})$ *such that* $P \notin \mathcal{DB}$*, the execution of* $Redeem(pk_{\mathcal{V}}, P, \mathcal{DB})$ *must return* `accept` *with overwhelming probabiliy.*
- *(Unforgeability) Receipt tokens* R *and loyalty points tokens* P *should be unforgeable against one-more forgery under chosen-message attacks; that is, for any integer* ℓ*, no polynomial adversary* $\mathcal{A}$ *should be able to successfully submit* $\ell + 1$ *receipt tokens after only* ℓ *executions of the* Buy *protocol, nor redeem* $\ell + 1$ *loyalty points tokens after only* ℓ *executions of the* Issue *protocol.*
- *(Anonymity) A vendor* $\mathcal{V}$ *should not be able to link a submitted receipt token* R *to the particular execution of the* Buy *protocol in which the token was produced. Likewise,* $\mathcal{V}$ *should not be able to link a submitted loyalty points token* P *to the particular execution of the* Issue *protocol in which the token was produced.*

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

- **Controlled linkability**. A customer $\mathcal{C}$ should be able to decide whether a submitted receipt token R can be linked to other receipt tokens submitted by the same $\mathcal{C}$ to the same vendor $\mathcal{V}$.
- **Untransferability**. A customer $\mathcal{C}$ should not be able to obtain loyalty points from $\mathcal{V}$ by submitting a purchase receipt issued to another customer $\mathcal{C}'$; transfer of purchase receipts among customers blurs their profiles and hence is against the vendor's interests. Note that transfer of purchase receipts among customers implies that the transferring customer loses the loyalty points she could get in exchange for that receipt, as submitted receipt tokens are invalidated. On top of that, the customer who transfers her receipt tokens loses some privacy w.r.t the customer to whom she transfers them. If losing loyalty points and partially losing one's privacy can be assumed sufficient to discourage the transfer of receipts, no specific countermeasures are needed to ensure untransferability. For the case where the above assumption does not hold, in Section 10 we

provide an extension of our protocol suite to enforce untransferability of both receipt and loyalty points tokens.

3 Related work

Certified tokens whose issuance and redemption are unlinkable, as we propose for receipt and loyalty points tokens, were first proposed in the anonymous electronic cash literature [13,14]. Anonymous e-cash typically relies on blind signatures [13], a type of signatures in which the signer does not learn the message she is signing. We provide a more detailed description of blind signatures (and also partially blind signatures, [4,5]) in Section 4. More recent works regarding anonymous electronic cash have used cryptographic accumulators [6] to easily trace malicious behaviors. Although we use techniques from anonymous electronic cash to build our scheme, we avoid using a trusted third party (a bank in the e-cash setting), because this could allow this third party to profile the customers.

More generally, secure multiparty computation (MPC), introduced in [36], allows a set of parties to engage in joint computation while keeping their inputs private. In this respect, blind signatures or partially blind signatures can be viewed as a special case of MPC. However, in most other applications of MPC, like privacy-preserving data mining [21] or electronic voting [29], participants giving their input are usually symmetric (they are peers with no hierarchy among them) and semi-honest (the inputs provided by participants are assumed to be valid ones). In the case of blind signatures and partially blind signatures, the above does not hold: i) there is asymmetry, because the party issuing blind signatures (bank issuing e-cash or vendor issuing receipts and loyalty points) has more power than the rest (the customers); ii) the parties seeking to obtain blind signatures may be malicious, in that they cannot be trusted to provide valid information (customers might submit banknotes with higher denomination than declared, or receipts of items that were not actually purchased).

In the previous two paragraphs, we have discussed work related to the privacy-preserving token issuance aspect of our contribution. If we now focus on its other main aspect, namely privacy-preserving customer profiling, there are certainly other works in which a service provider collects information on her user base in a privacy-preserving way. For example, the authors from [28] propose a scheme to collect location-based aggregate statistics. In that contribution, drivers provide location-based information (*e.g.* their speed), to a centralized server that can aggregate this information. The information they provide is always encrypted, using some homomorphic encryption scheme. The data they deal with is only numeric, and the computations that can be done are defined beforehand (they depend on the homomorphic encryption scheme in use). In our contribution, however, data do not need to be numeric, and we protect them by generalization rather than encryption; note that, unlike homomorphic encryption, generalization does not restrict computations that can be carried out on the protected data.

4 Cryptographic background

In this section we summarize the cryptographic background we use in our proposed mechanism. Specifically, we recall asymmetric bilinear maps, partially blind signatures and zero-knowledge proofs. The latter are relevant to the untrasferability property in Section 10. We recall formal security properties when relevant and we set the notation for the rest of the paper.

4.1 Bilinear maps

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

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

We use multiplicative notation for all groups $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$.

4.2 Partially blind signatures

Blind signature protocols are interactive protocols between a user, a signer and a verifier, in which the signer produces a digital signature of a message submitted by the user, but does not learn anything about the contents of the message. This primitive was introduced by Chaum in [13] and has since been used in a vast array of privacy-related protocols, such as e-cash, electronic voting and anonymous credential systems. An inherent drawback of blind signature protocols is that the signer cannot enforce a certain format on the message. Traditionally, this problem has been solved using cut-and-choose techniques, in which the requester of a signature generates and blinds a number n of messages, the signer asks the requester to unblind all messages but a randomly chosen one, checks whether all unblinded messages conform to the required format and, if yes, signs the only message that remains blinded. Using cut-and-choose techniques solves the problem (the probability that the requester succeeds in getting a non-conforming message signed is upper-bounded by $1/n$), but it does so at the cost of high computation and communication overheads.

Partially blind signatures were introduced by Abe in [4,5] as an alternative to cut-and-choose protocols. In a partially blind signature protocol, the user and the signer agree on a public information that is to be included in the signed message. Both the user and the signer can be sure that such information is really included in the signature, and the secret message of the user remains blinded to the signer.

Definition 3. *(Partially Blind Signature scheme [38]) A partially blind signature consists of three participants: signer, user and verifier. There are three algorithms: Key Generation, Partially Blind Signature Issuance, and Verification.*

- *Key Generation is a probabilistic polynomial-time algorithm that takes security parameter γ and outputs a key pair (pk, sk).*
- *Partially Blind Signature Issuance is an interactive protocol between the signer and the user. The public inputs of both the user and signer contain the previously agreed upon public information* info*. The private input of the signer is sk, and the private input of the user is the message m to be signed. Upon protocol completion, the private output of the user contains either* `fail` *or* $(\mathsf{info}, m, \sigma)$*, where σ is the signer's signature on* info *and m.*
- *Verification is a polynomial-time algorithm that takes $(pk, \mathsf{info}, m, \sigma)$ as input and outputs either* `accept` *or* `reject`*.*

Note that, in our loyalty program setting, the signer and the verifier will typically be the same entity (the vendor). However, the user can also verify the validity of the signature at any moment.

Definition 4. *(Correctness and security of a partially blind signature scheme) A partially blind signature scheme is considered correct and secure if it satisfies the following properties:*

- *(Correctness) For honestly generated parameters, and for honest execution of Partial Blind Signature Issuance, the resulting partial blind signature passes verification with overwhelming probability.*
- *(Partial blindness) The following must hold: 1) the signer must be assured that the embedded public information is included in the signature, and that no one can modify this information; and 2) based on the embedded information, a signer cannot link the signature with the concrete issuing instance that produced it.*
- *(Unforgeability) A partially blind signature scheme is called unforgeable*

against one-more forgery under chosen-message attack if, for some integer ℓ, and a given public information info*, there is no probabilistic polynomial-time adversary $\mathcal{A}$ that can compute, after ℓ interactions with the signer,* $\{(\mathsf{info}, m_j, \sigma_j)\}_{j=1,\dots,\ell+1}$ *valid signatures with non-negligible probability.*

Boldyreva proposed in [8] a blind version of the BLS signature scheme from [10]. The security of Boldyreva's scheme rests on the chosen-target version of the Computational Diffie-Hellman problem in Gap Diffie-Hellman groups. In [37], Zhang *et al.* proposed a signature scheme based on the $k + 1$ Exponent Problem. Later, the same authors published in [38] a blind version of the previous scheme, following the same approach of Boldyreva.

We use a partially blind signature scheme from bilinear pairings presented in [38]. This scheme satisfies the requirements of correctness, partial blindness and unforgeability against one-more forgery under chosen-message attacks in the random oracle model under the inverse CDH assumption in bilinear groups, and thus it is considered secure. Security proofs can be found in [38]. Additionally, this scheme produces short signatures, it is computationally efficient and allows aggregate verification of signed messages bearing the same agreed public information. Note that we use this partially blind signature scheme as a black box, and any other secure scheme would fit in our proposed protocol. For completeness, we describe this scheme in the following sections.

4.3 Zero-knowledge proofs

A zero-knowledge proof (ZKP), as introduced by Goldwasser *et al.* in [19], is a method whereby a party (the prover) can prove to another party (the verifier) that a given statement is true, without leaking any information beyond the fact that the statement is true.

In [23], the author introduces a general framework based on one-way homomorphisms to prove in zero-knowledge the knowledge of the preimage of a group homomorphism. He shows that the Schnorr protocol [30] is an instance of this general framework, using the exponentiation in groups in which the discrete logarithm problem is hard. Our construction uses a variation of this protocol, based on the exponentiation in bilinear groups in which the discrete logarithm problem is assumed hard.

5 Generalization of purchase histories

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

To generalize receipts, a vendor must use a publicly available taxonomy for the products she offers. This taxonomy $\mathcal{T}$ is modeled as a tree, being its root node a generic identifier such as *Product*, and each leaf a specific product in the set of products $P = \{p_1, \ldots, p_n\}$ on sale. The inner nodes of the tree are the subsequent categories to which the products belong: the closer to the leaf nodes, the more specific categories are.

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

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

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

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

6 Anonymous tokens with controlled linkability

As stated in Section 2, loyalty points and purchase receipts have requirements in line with those of anonymous electronic cash and anonymous electronic credentials. These well-known primitives commonly use blind signatures and/or zero-knowledge proofs of knowledge [11,12,14,26]. We will treat loyalty points and purchase receipts using a construction that we call anonymous tokens with controlled linkability. These tokens will be realized by using a partially blind signature construction, namely the one described in [38], with slight changes introduced in the messages to be signed.

6.1 Controlled linkability of tokens

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

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

6.2 Description

Anonymous tokens with controlled linkability are operated in four phases:

- In the *setup* phase, a certification agency generates the public parameters of the partially blind signature scheme.

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

6.2.1 Setup

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

6.2.2 Key generation

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

6.2.3 Token issuance

A customer wants to obtain from a vendor a token with an agreed public information c (this information may specify a number of loyalty points or a purchase receipt for a certain product). This is an interactive protocol which produces a partially blind signature on public information c, and a secret message containing a unique identifier α of the token and a (possibly) unique identifier y. The protocol is depicted in Figure 1 and described next:

(1) The customer chooses a value for y, either from a list of previously used values or by generating a new one uniformly at random from $\mathbb{Z}_q^*$.
(2) The customer and the vendor agree on a public string $c \in \{0,1\}^*$.
(3) The customer chooses random $\alpha, r \in_R \mathbb{Z}_q^*$ and builds the message $m = (\alpha, y)$. Then, the customer blinds the message by computing $u = H_0(c||m)^r$ and sends u to the vendor.
(4) The vendor signs the blinded message by computing $v = u^{(H(c)+sk_\mathcal{V})^{-1}}$ and sends it back to the customer.

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

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

$$
\begin{array}{ccc}
\textbf{Customer} & & \textbf{Vendor} \\
\hline
y \in \mathbb{Z}_q^* & & sk_{\mathcal{V}} \\
 & \xleftrightarrow{c \in \{0,1\}^*} & \\
\alpha, r \in_R \mathbb{Z}_q^* & & \\
m = (\alpha, y) & & \\
u = H_0(c||m)^r & \xrightarrow{\quad u \quad} & \\
 & \xleftarrow{\quad v \quad} & v = u^{(H(c)+sk_{\mathcal{V}})^{-1}} \\
\sigma = v^{r^{-1}} & & \\
T = \langle c, m, \sigma \rangle & &
\end{array}
$$

Figure 1. $\mathsf{Issuance}$ protocol

6.2.4 Token verification

The submission and verification of a token is an interactive protocol between a customer and a vendor. The customer submits the token $T = \langle c, m, \sigma \rangle$ and the vendor returns `accept` or `reject` as a result of the verification. Informally, the vendor checks that the signature on the token is valid and has been produced by himself; then, if the value y contained in the message matches the one of a previously submitted token, the tokens are grouped. The protocol is outlined in Figure 2 and described next:

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

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

If the above equality holds, he checks whether the token has already been spent (he verifies whether a token with the same α has previously been submitted). If the verification was successful and the token has not been spent yet, he marks it as spent and sends an `accept` message to the customer. Otherwise, he sends a `reject` message to the customer.

(4) Finally, the vendor checks whether the identifier value y is the same as the one in a previously spent token. If yes, he links the new token with that previous one.

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

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

Figure 2. Verification protocol

6.2.5 *Aggregate verification*

This protocol allows the customer to aggregate signatures of messages bearing the same public information by just multiplying the resulting signatures. If there is a list of tokens $\{T_1, \ldots, T_n\}$, where $T_i = \langle c_i, m_i, \sigma_i\rangle$, and $c_i = c$ for $1 \leq i \leq n$, a customer can aggregate the partially blind signatures by computing $\sigma_{agg} = \prod_{i=1}^{n} \sigma_i$ and submitting $T_{agg} = \langle c, \{m_1, \cdots, m_n\}, \sigma_{agg}\rangle$. The vendor can then verify the validity of the aggregated token by checking the equality

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

7 Privacy-aware loyalty program construction

Our proposed solution for privacy-aware loyalty programs builds on the anonymous tokens with controlled linkability we described in Section 6 and the generalization of purchase histories described in Section 5. As introduced in Section 2, our construction consists of the following protocols: Setup, VendorSetup, Enroll, Buy, Submit, Issue and Redeem.

7.1 Setup

The setup phase is run by a certification authority to generate the public parameters params of the anonymous token with controlled linkability construction described in Section 6. These parameters are made public to every $\mathcal{V}$ offering loyalty programs and to every $\mathcal{C}$ intending to participate in them.

7.2 VendorSetup

Each vendor $\mathcal{V}$ publishes a product taxonomy $\mathcal{T}_{\mathcal{V}}$ as described in Section 5. Then, $\mathcal{V}$ obtains a key pair built as described in the key generation procedure in Section 6. Finally, $\mathcal{V}$ publishes his public key.

7.3 Enroll

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

7.4 Buy

A customer $\mathcal{C}$ in a loyalty program offered by a vendor $\mathcal{V}$ purchases a product, either at a physical or online store of $\mathcal{V}$. Note that, in the case of an online store, $\mathcal{C}$ should use additional anonymization measures, such as anonymous Internet surfing, offered for example by Tor networks [32], anonymous shipping methods [20], and anonymous payment methods (*e.g.* [25,14,24] or simply prepaid scratch cards). The protocol is as follows:

(1) $\mathcal{C}$ sends to $\mathcal{V}$ the name p_i of the product $\mathcal{C}$ wants to buy.
(2) $\mathcal{C}$ chooses a value y to be used in the token issuance protocol, depending on her privacy preferences: if she wants the new purchase receipt to be linkable to previously obtained purchase receipts (linkability is incentivized as described in Section 7.8 below), she will reuse the same y that was used in those previous receipts; if she does not want this new purchase receipts to be linkable to previous receipts, she will pick a new random $y \in \mathbb{Z}_q^*$.

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

7.5 Submit

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

Additionally, as said in Section 5, for each product she also submits all tokens from the selected generalization level up to the root of the taxonomy (to ensure tokens in the generalization path cannot later be used as independent purchase receipts). Submission of each token T_i is performed according to the $\mathsf{Verification}(\mathcal{V}, \mathcal{P}, R_i)$ protocol described in Section 6.2.

7.6 Issue

To issue loyalty points, the vendor builds a message info that encodes an identifier of the vendor, the number of points this token is worth and an expiration date. Unlike for purchase receipts, the vendor has no legitimate interest in linking several tokens containing loyalty points and thus does not incentivize linkability. Hence, the customer picks a fresh random y for each new loyalty points token she claims. Then the vendor and the customer run the interactive protocol $\mathsf{Issuance}(\mathcal{V}, \mathcal{P}, \mathsf{info}, y)$. The generated token contains the loyalty points issued to the customer.

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

7.7 Redeem

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

It is possible to simultaneously redeem several loyalty points tokens by using the aggregation of signatures described in Section 6.2.

7.8 Incentives related to purchase receipts submission

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

- To encourage customers to use little or no purchase receipt generalization (and hence to renounce some of their privacy), the amount of loyalty points awarded per receipt token should depend on the chosen level of generalization: more loyalty points awarded to less generalized purchase receipts.
- If the customer submits unlinkable receipts, she should just get enough loyalty points to reward her as a returning customer. To encourage customers to allow linkage of purchase receipt tokens by the vendor (and hence customer profiling), a customer should get more loyalty points if she submits $n_1 + n_2$ tokens with the same y value than if she submits n_1 tokens with one y value and then n_2 tokens with a different y value (*superlinear reward*). Furthermore, the vendor may require that the list of linkable receipt tokens for which reward is claimed correspond to purchases made within a certain time window (if linking purchases very distant in time is uninteresting for profiling).
- Two or more customers might be tempted to share their y values in order to submit a longer list of linkable receipts and thereafter share the superlinear number of loyalty points they would earn. As long the reward is only *slightly* superlinear, customer collusion is discouraged if the customer $\mathcal{C}$ who submits the list of linkable tokens is required by $\mathcal{V}$ to actually show all the actual linkable tokens (and not just a reference to them): colluders different from $\mathcal{C}$ may not like to pay the privacy toll of disclosing their purchase receipts to $\mathcal{C}$.

8 Complexity and security analysis

We count here the number of operations required by the Issuance and Verification protocols described in Section 6. These are the two performance-critical protocols, because they are the ones that need to be run every time a token (carrying a purchase receipt or loyalty points) is to be generated or verified.

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

As shown by the computing times given in Section 9 below, the above computations can be done in a very reasonable time, even though the customer's computation must be carried out by her smartphone.

Regarding security, we justify here that, except for the property of untransferability, which will be dealt with in Section 10, the remaining security requirements identified in Section 2.2 are satisfied by the above protocol suite.

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

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

- **Unforgeability.** Unforgeability against one-time forgery under chosen message attack of receipt and loyalty points tokens is provided by the partially blind signature scheme. Note that we use the partially blind signature recalled in Section 4.2 as a black box: the Issuance and Verification protocols in Section 6.2.3 and 6.2.4, respectively, use partially blind signatures *exactly* as described in [38]; hence, the security proof of [38] guarantees the security of such protocols.
- **Anonymity.** No information on the user is obtained by a server during the protocol. Submitted tokens cannot be linked to issued tokens or to the identity of a requester or prover because of the *partial blindness* property of the signature scheme [38]. The justification of partial blindness given in [38] is briefly recalled in the Appendix.
- **Controlled linkability.** When a token is issued, the identifying value y

is only known to the customer who generated the token, due to the *partial blindness* of the signature. Hence, if two verified tokens contain the same identifying value y, there are two possibilities: i) both tokens were generated by the same customer, who reused y to allow the vendor to link them; ii) the customer who generated one token leaked y to the customer who generated the other token. If the latter leakage is prevented by technical means or discouraged with appropriate incentives (see discussion in Section 7.8), then two tokens containing the same y can be linked by the vendor as corresponding to the same customer.

9 Experimental results

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

9.1 Execution times from a prototype

We created a testbed to test the performance of the new protocol in a real environment. Specifically, we wrote a prototype Android customer's application that requests and submits tokens and a vendor application running on a laptop that issues and verifies tokens. To conform to current loyalty program implementations, the communication between customer and vendor is done via NFC using Android's host-card emulation (HCE). Generalization of purchase receipts is done using the Walmart Open API [35] which, on input a product name, returns a list of categories to which the product belongs. The cryptographic protocol was implemented using the jPBC library version 2.0 [16], which runs both in standard and Android Java without further dependencies.

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

The first step in a purchase receipt issuance is to obtain the generalization path of the product, from the product name up to the root of the product taxonomy. Using Walmart Open API requests, the mean response time was

1,534 ms. This step does not affect the issuance of loyalty points, and can be drastically reduced if caching mechanisms are used or if the product taxonomy is stored locally. To test the impact of caching, we implemented a naive caching mechanism, by which responses from the Walmart Open API are stored in a hash table together with the corresponding query. When the vendor makes a query, we check if it was made previously; if yes, we obtain the response from the hash table instead of sending the query to the online service. The tests have shown that the time to obtain the generalization path of a previously queried product name when using caching is reduced to 229 nanoseconds for a universe of 6,167 different products. Clearly, that is a more than affordable time.

Figure 3 shows the execution time for the Issuance protocol, broken down by the different stages. The measured total time is 659.1 ms, divided in three stages: blinding, executed by the client application on the customer's smartphone; signing, executed by the vendor on the laptop; and unblinding, again executed by the application on the smartphone. The three stages take, respectively, 356.4 ms, 19.5 ms and 141.7 ms. The difference between the total measured time and the sum of the three stages is 141.5 ms, which corresponds to communication overheads.

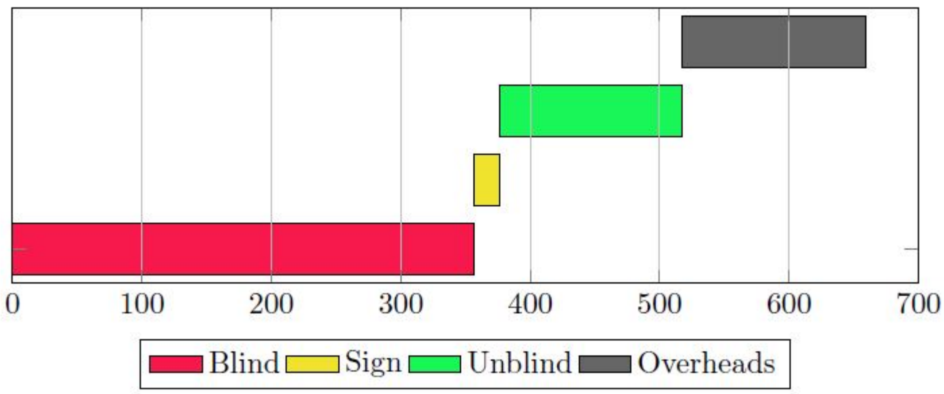

Figure 3. Execution times for the Issuance protocol

Figure 4 shows the execution time for the Verification protocol. The measured execution time to verify the validity of a token by the vendor is 275.7 ms, while the remaining 401.7 ms is the communication overhead due to the transmission of the whole token (which does not fit in a single APDU).

9.2 Deployability analysis in stores

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

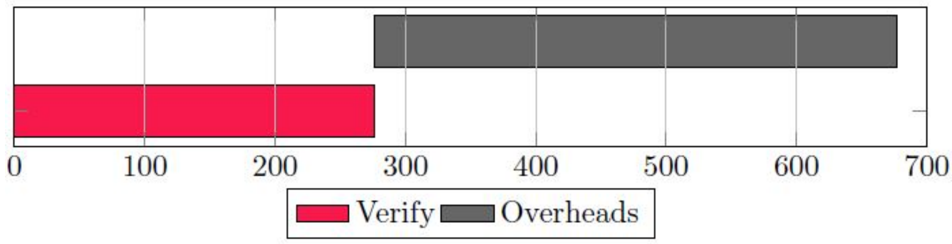

Figure 4. Execution times for the Verification protocol

9.2.1 Physical store

We model the impact of the application of our solution in physical stores by using queueing theory [18]. By comparing the typical perfomance of the queues at a store with the performance when our solution is applied, we can estimate the additional resources (number of queues) that the store would need to cope with the additional workload incurred by our solution.

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

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

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

Section 9.1). Since token issuance takes place in parallel to product checking, the expected service time is $1/\mu = \max(2, 0.65d)n + 30$. Finally, we assume the number of cashiers in a shop is $q = 4$ (which is a rather modest estimate for large stores such as Tesco's).

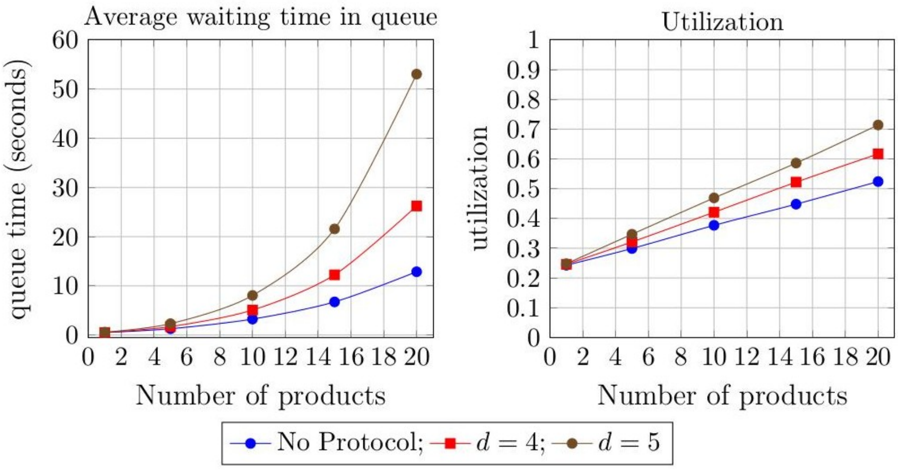

Figure 5. Mean waiting time and utilization of cashiers, depending on the number of products per customer and the generalization depth d

In Figure 5, we show the mean waiting time at each of the queues/cashiers and the utilization of each of the cashiers as a function of the average number of items in the customers' baskets and the depth of the generalization tree. The blue curve corresponds to three situations: not using the protocol (just bare product checking without token issuance) and also using the protocol with $d = 2, 3$. We label it 'No Protocol' because, even if the protocol is used with depths 2 or 3, the 2 seconds needed to check each product dominate the time to issue d tokens. We see that the utilization reaches 70% with 20 products and $d = 5$. Although this is acceptable, adding one more queue/cashier might be advisable for those parameter values.

Analytically, assuming an M/M/q model [18], for token issuance time τ, average number n of products per customer, depth d of the generalization tree, expected inter-arrival time $1/\lambda$ and number of cashiers q, the utilization of each cashier is

$$\rho = \frac{\lambda}{\mu q} = \frac{\lambda(\tau n d + c)}{q}, \tag{1}$$

and the mean waiting time is

$$T_w = \frac{C(q, \lambda/\mu)}{\mu q - \lambda} = \left(\frac{\tau n d + c}{q}\right)\left(\frac{C(q, \lambda(\tau n d + c))}{1 - \rho}\right), \tag{2}$$

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

Results in [31] show that the mean waiting times in a scenario with parallel queues and jockeying (switching queues), which is the typical scenario at a store, are similar (but not equal) to an M/M/q scenario. Formula (2), then, gives no more than an approximation to the actual mean waiting time.

Expressions (1) and (2) are helpful to design a store. For example, given values of τ, n, c and λ for a certain store, and given maximum acceptable values of ρ and T_w (imagine the maximum for ρ is specified by unions and the maximum for T_w by a consumer survey), the store designer can determine the necessary number of cashiers q if using a generalization hierarchy with depth d is desired.

9.2.2 Online store scenario

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

Our implementation proposal is as follows. Customers adding products to the cart immediately compute the blinding phase of the receipt token issuance for the selected product and all its generalizations. For tree depth d, this step takes $356.4d$ ms of computation by the customer (time according to our prototype described in Section 9.1). The online store adds the product to the cart database, as well as the cryptographic material sent by the user. Optionally, the store could execute the signing phase of the token issuing protocol at this moment, which takes $19.5d$ ms (again according to our prototype). While signing tokens in parallel to product checking has the advantage of making checkout lighter, it might be a waste of resources to sign a receipt token of a product the customer has not yet paid for (because the customer might decide to withdraw one or several products from her basket before checkout). When the customer finally checks out, the store computes (if it has not done so previously) the signature on all receipt tokens and sends them to the customer. At this point, the transaction is already over. The customer can then unblind the signatures on her tokens offline. The additional storage capacity needed by the store is d times the storage required by a token signature per product entry in the cart database ($512d$ bits in our prototype).

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

Both in a physical and in an online store, the issuance of loyalty points tokens can be analyzed in a way similar to the issuance of receipt tokens, except that loyalty points do not depend on the d factor (whereas d receipt tokens are always issued per product, loyalty points tokens can be issued one at a time). Otherwise, the computation and the storage required to issue one loyalty points token are the same as for one receipt token. Spending loyalty points would be done in the checkout phase, and this would increase the computation time by 275.7 ms per token (the verification time by the store, according to our prototype described in Section 9.1).

10 Untransferability of tokens

As mentioned in Sections 2.2 and 7.8 above, losing loyalty points and purchase privacy is often enough to deter a customer from giving her receipts to another customer. However, in some cases additional deterrence by $\mathcal{V}$ may be needed to prevent purchase receipt transfer and even loyalty points transfer among customers. In this section we propose an extension of our token issuance and verification protocols to enforce untransferability of tokens.

Note that the vendor may be interested in preventing transfer only for one type of tokens. Untransferability is more critical for purchase receipts (because their transfer prevents customer profiling), whereas transferring loyalty points among customers may be tolerable in many situations. If only one type of tokens needs to be made untransferable, then the extended issuance and verification protocols described in this section should only be applied to that type of tokens.

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

The commitment keys should be securely stored in the device to prevent users from transferring them. If this is achieved, then the tokens are also untransferable. To do so, one might use a trusted platform module installed in the device. For example, Google's Nexus series (up to Nexus 4) carry a TPM integrated in the NFC chip. A more practical solution, and one which is currently gaining popularity among banking applications (and credit card emulation apps) is to use privileged instructions of the device's main processor, a standard ARM extension called TrustZone [33]. Making use of this feature, Android provides a hardware-backed secure keystore API which allows applications to store certificates and keys in a secure way. A key stored by some application can only be retrieved by the same application, even if the device has been compromised. This approach has been used for secure applications such as secure PIN entry, digital rights management, e-ticketing, etc. KNOX [3] is a technology from Samsung, which also uses the TrustZone extension to provide a secure execution environment in mobile devices.

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

- Both the setup and the key generation phases remain mostly as described in Sections 6.2.1 and 6.2.2 above, respectively. However, during key generation an additional *commitment key* $sk_{\mathcal{C}} \in_R \mathbb{Z}_q^*$ is generated and given to the customer $\mathcal{C}$ in a secure device (*e.g.* a smart card) such that the key cannot be modified or extracted from the device.
- More changes are needed in the Issuance and Verification protocols with respect to Sections 6.2.3 and 6.2.4, respectively. We describe these changes in the following subsections.

10.1 *Extended* Issuance *protocol*

During this phase, the customer $\mathcal{C}$ commits to the public and the secret information contained in the token with her commitment key, as in Schnorr's ZKP protocol [30]. Namely, Step (3) in Section 6.2.3 is replaced by the following one:

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

The rest of the protocol of Section 6.2.3 remains unaltered. The resulting extended protocol including this modification is shown in Figure 6.

10.2 *Extended* Verification *protocol*

During this phase, the vendor checks whether the signature on token T was correctly computed (normally by the vendor himself at a previous time) and whether the token has not yet been spent. Additionally, this version of the protocol includes Schnorr's 3-step interactive ZKP [30] to prove knowledge of the commitment key $sk_{\mathcal{C}}$.

First, and to account for the changes in the contents of the token, replace Step 2 of the Protocol in Section 6.2.4 by the following step:

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

Customer		**Vendor**
$sk_{\mathcal{C}}, y \in \mathbb{Z}_q^*$		$sk_{\mathcal{V}}$
	$\xleftrightarrow{c \in \{0,1\}^*}$	
$\alpha, r \in_R \mathbb{Z}_q^*$		
$h = H_0(c\|\|\alpha\|\|y)$		
$m = (\alpha, y, h^{sk_{\mathcal{C}}})$		
$u = H_0(c\|\|m)^r$	$\xrightarrow{u}$	
	$\xleftarrow{v}$	$v = u^{(H(c)+sk_{\mathcal{V}})^{-1}}$
$\sigma = v^{r^{-1}}$		
$T = \langle c, m, \sigma \rangle$		

Figure 6. Issuance protocol with untransferability

Then, if the signature verification at Step 3 is correct and the token has not yet been spent, notify the customer to start the interactive ZKP to prove knowledge of the commitment key. This implies adding the following steps between Steps 3 and 4 of the original protocol

(3a) The customer chooses $k \in_R \mathbb{Z}_q^*$ and sends $t = h^k$ to the vendor.
(3b) The vendor answers with a challenge $sc \in_R \mathbb{Z}_q^*$.
(3c) The customer computes a response $r = k + sk_{\mathcal{C}} \cdot sc$ and sends r to the vendor.
(3d) If $h'^r \stackrel{?}{=} t \cdot h^{sk_{\mathcal{C}} \cdot sc}$ holds, the vendor sends an `accept` message to the customer. Otherwise, the vendor sends a `reject` message.

If the customer can prove knowledge of the commitment key, the vendor is convinced that she participated in the issuance of the token. Hence, the token has not been transferred to a different customer since it was issued. The resulting extended protocol is shown in Figure 7.

10.3 Complexity and security analysis of the extension

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

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

Figure 7. Verification protocol with untransferability

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

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

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

The security of the extension is based on:

- The security of Schnorr's ZKP, proven in [30,23]. Hence, a customer not holding the commitment key cannot convince a vendor that she holds the key. Also, neither the vendor nor any observer learn any bit of the commitment key.
- The tamper-resistance of the secure device. This device performs all customer computations involving the commitment key, in such a way that

the customer cannot learn the value of her own key. This ensures that no customer can transfer her commitment key to another customer. Hence, when a customer proves during token verification that she holds the same commitment key used for token issuance, the vendor is convinced that the token has not been transferred.

11 Conclusions

In our privacy-aware alternative to traditional loyalty programs, the customers are granted the power to decide what private information they want to disclose, and how accurate that information is. We have described a privacy-aware protocol suite that still offers the two main features of loyalty programs: to reward returning customers and to make customer profiling possible. We have presented experimental results that show that our suite is usable in practice. Finally, we have proposed an extension of our protocol suite which prevents purchase receipts and loyalty points from being transferred among customers; if only transfer of one type of tokens is to be prevented, then our extension need only be used to issue and verify that type of tokens.

Acknowledgments

We thank Dr Qiang Tang for useful discussions on an earlier version of this paper. We also appreciate the insightful comments by the reviewers. The following funding sources are acknowledged: Google (Faculty Research Award to the second author), Government of Catalonia (ICREA Acadèmia Prize to the second author and grant 2014 SGR 537), Spanish Government (projects TIN2011-27076-C03-01 "CO-PRIVACY" and TIN2014-57364-C2-1-R "SmartGlacis"), European Commission (project H2020-644024 "CLARUS") and Templeton World Charity Foundation (grant TWCF0095/AB60 "Co-Utility"). 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 Google, UNESCO or the Templeton World Charity Foundation.

References

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

[2] "One million fewer customer visits a week at Tesco", *The Guardian*, June 3, 2014. http://www.theguardian.com/business/2014/jun/03/tesco-morrisons-sales-fall-further-aldi-lidl

[3] "White Paper: An Overview of the Samsung KNOX Platform," accessed November 27, 2015.
http://www.samsung.com/uk/business-images/insights/2015/An_Overview_of_the_Samsung_KNOX_Platform_V1.11-0.pdf

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

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

[6] M.H. Au, Q. Wu, W. Susilo, Y. Mu, "Compact e-cash from bounded accumulator," in *Topics in Cryptology–CT-RSA 2007*, LNCS 4377, pp. 178–195, Springer, 2006.

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

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

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

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

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

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

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

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

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

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

[17] C. Dunn, "Loyalty programs and privacy issues: do you need to worry about providing personal information?" *Disney Family*, accessed November 27, 2015. `family.go.com`

[18] N. Gautam, *Analysis of Queues: Methods and Application*, CRC Press, 2012.

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

[20] R.C. Johnson, "eDropship: methods and systems for anonymous eCommerce shipment," *US Patent 7,853,481*, 2010.

[21] Y. Lindell and B. Pinkas, "Privacy-preserving data mining," in *Advances in Cryptology–CRYPTO 2000*, LNCS 1880, pp. 36–54, Springer, 2000.

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

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

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

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

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

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

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

[29] A. Riera, *Design of Implementable Solutions for Large Scale Electronic Voting Schemes*, PhD Thesis, Autonomous University of Barcelona, 1999.

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

[31] A.M.K. Tarabia, "Analysis of two queues in parallel with jockeying and restricted capacities," *Applied Mathematical Modelling*, 32(5):802–810, 2008.

[32] "Tor Project: Anonymity Online," accessed November 27, 2015. www.torproject.org

[33] "TrustZone - ARM," accessed November 27, 2015. www.arm.com/products/processors/technologies/trustzone/

[34] B. Tuttle, "A disloyalty movement? Supermarkets and customers drop loyalty card programs," *TIME*, 2013. http://business.time.com/2013/07/11/a-disloyalty-movement-supermarkets-and-customers-drop-loyalty-card-programs/

[35] Walmart Open API. https://developer.walmartlabs.com/

[36] A. C. Yao, "Protocols for secure computations (extended abstract)," in *Foundations of Computer Security–FOCS*, pp. 160–164, IEEE Computer Society, 1982.

[37] F. Zhang, R. Safavi-Naini, W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," in *Public Key Cryptography–PKC 2004*, LNCS 2947, pp. 277–290, Springer, 2004.

[38] F. Zhang, R. Safavi-Naini, W. Susilo, "Efficient verifiably encrypted signature and partially blind signature from bilinear pairings," in *Progress in Cryptology–INDOCRYPT 2003*, LNCS 2904, pp. 191–204, Springer, 2003.

A Partial blindness

In the blinding phase of the so-called partially blind signature of [38], r is randomly chosen from $\mathbb{Z}_q^*$, and thus $u = H_0(c||m)^r$ is a random element of the group $\mathbb{G}_2$. The signer receives this random information and the public information which he already knows. Therefore, no information about the message is leaked.

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

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

or equivalently

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

Verification (A.1) would only pass for $c' \neq c$ if $H(c') = H(c)$ and $H_0(c'||m) = H_0(c||m)$. This is unlikely, because H, H_0 are cryptographic hash functions.

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

Hence the so-called partially blind signature scheme of [38] really satisfies partial blindness.