inShopnito: An Advanced yet Privacy-Friendly Mobile Shopping Application

Andreas Put, Italo Dacosta, Milica Milutinovic, Bart De Decker
KU Leuven, Dept. of Computer Science, iMinds-DistriNet
Celestijnenlaan 200A, 3001 Heverlee, Belgium
firstname.lastname@cs.kuleuven.be

Stefaan Seys
KU Leuven, Dept. of Electrical Engineering, iMinds-Cosic
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

Faysal Boukayoua, Vincent Naessens
KU Leuven, Dept. of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Gent, Belgium
firstname.lastname@kahosl.be

Kris Vanhecke, Toon De Pessemier, Luc Martens
Ghent University, iMinds-WiCa
Gaston Crommenlaan 8, 9050 Ghent, Belgium
firstname.lastname@intec.ugent.be

***Abstract*—Mobile Shopping Applications (MSAs) are rapidly gaining popularity. They enhance the shopping experience, for instance by offering customized recommendations or incorporating customer loyalty programs. Although MSAs are quite effective at attracting new customers and binding existing ones to a retailer's services, existing MSAs have several shortcomings. The data collection practices involved in MSAs and the lack of transparency thereof are important concerns for many, increasingly privacy-aware, (potential) customers. This paper presents inShopnito, a privacy-preserving mobile shopping application. All transactions made in inShopnito are unlinkable and anonymous. However, the system still offers all features expected from a modern MSA. Customers can take part in loyalty programs and earn or spend loyalty points and electronic vouchers. Furthermore, the MSA can suggest personalized recommendations even though the retailer cannot construct rich customer profiles. These profiles are managed on the smartphone and can be partially disclosed in order to get better, customized recommendations. Finally, we present an implementation of inShopnito, the security and performance of which is analyzed. In doing so, we show that inShopnito combines multiple advanced technologies into a secure, privacy-preserving and practical mobile shopping application.**

***Keywords*-Software Engineering; Security; Privacy**

I. Introduction

In recent years, mobile shopping applications (MSAs) have made a significant rise in popularity. MSAs are (usually smartphone-based) applications that provide customers with an enhanced shopping experience. They are cheap to deploy, as most people carry smartphones and are already familiar with mobile applications. One important feature of an MSA is to facilitate the participation in Customer Loyalty Programs (CLPs) [1]–[4]. Customers enrolled in a CLP receive different incentives (e.g., loyalty points, discount vouchers, gifts, etc.) on each transaction with the retailer. They can collect *loyalty points* that can later be exchanged for discounts or promotional items. CLPs allow providers not only to encourage loyal behaviour but also to collect and analyze data about customers (e.g., brand preferences, average spending, etc.). This information is useful to retailers for planning better marketing and advertising strategies. In addition, MSAs can provide more advanced services. For instance, Belgacom and BNP Paribas Fortis recently announced the *Belgian Mobile Wallet* [5], a mobile application that, in addition to loyalty cards, will support mobile identity management and payments. Furthermore, MSAs can enhance the shopping experience by providing relevant recommendations to its users.

Unfortunately, existing MSAs have major shortcomings that hinder their complete adoption. Current data collection practices raise privacy concerns among customers. For instance, a recent survey [6] shows that nearly 30% of the participants think that CLPs require too much personal information and 24% indicated that privacy concerns are the reason why they do not enroll in such programs. These privacy concerns have given CLPs a negative reputation [7]–[9]. In addition, this problem is even worse for the smartphone-based MSAs because they can use the customer's smartphone to gather additional information, like the current geolocation or the customer's actual identity. Finally, although the collected information allows retailers to send customized and more accurate recommendations to the MSA, it is unclear what data exactly is collected by the MSA, and what retailers actually do with the data (e.g., sell it to a third party).

In this paper we present inShopnito, a privacy-friendly mobile shopping application. inShopnito incorporates a series of security- and privacy-enhancing technologies into a full-fledged MSA. Retailers can reward customers with anonymous and unlinkable loyalty points and electronic vouchers. In addition, transactional data is securely stored and managed on the smartphone itself, resulting in local profiles controlled by customers. Users can opt-in to disclose parts of their (anonymized) profile in order to get customized recommendations from a recommendation system, without sacrificing their privacy. Furthermore, the integrity of the revealed information is protected by inShopnito and, therefore, the data collected by providers is not only less

sensitive, but also more accurate. In addition, inShopnito could be used to combine CLPs from multiple providers, thus, facilitating cooperation. Finally, the system facilitates retailers to comply with privacy laws because the data that is being stored is less sensitive.

In short, this paper makes the following key contributions:

- **Design of a privacy-protecting and practical mobile shopping application** We present inShopnito, a privacy-protecting MSA which offers many of the same benefits for customers as well as retailers compared to the traditional approach. Although a retailer cannot monetize the collected data by reselling it, privacy-concerned customers will be more inclined to use the service and, hence, the services of the retailer.
- **Implementation and evaluation of our proposed solution** We implemented the inShopnito system and created an Android application to demonstrate its feasibility. The performance results of the main user-actions (e.g., authentication, checkout, application start-up time) show that the system is practical to use.

II. Related Work

Even though privacy is recognized as an important issue in the loyalty schemes [10], [11], there is not a significant body of research addressing this issue. The existing proposals for privacy in the loyalty schemes usually focus on unlinkability of the loyalty points issued to the user. For instance, in the proposal by Enzmann and Schneider [12], blind signatures are utilized to avoid linking loyalty points in the issuing and the spending interaction with the provider. However, their proposed solution cannot securely prevent the users from sharing their loyalty points. On the other hand, in [13] the authors suggest that users anonymously download a loyalty card from a large pool to assure their privacy. Although the users remain anonymous towards the provider, their loyalty cannot be measured. A common issue for these two approaches is that the retailer cannot record any data, even with user consent. The shopping data that is collected through the loyalty system is also employed for determining the interests and for offering relevant products to the user. Work by Hardt and Nath [14] allows users to choose the amount of personal information that is to be disclosed to an ad server in order to be offered personalized ads. The user sends (a part of) personal preferences to the ad server, which are used for coarse-grained filtering of ads that are sent to user's device. The user application then performs the final filtering of the received ads based on the stored details on the user preferences. Similarly, we allow the user to be presented with recommendations based on the current shopping cart contents. In this work, we also enable the user to collect loyalty points and subsequently prove ownership of them without the two interactions being linkable. The practicality of this proposal is improved compared to existing solutions, as it allows users to utilise one application to manage all aspects of the shopping interactions.

III. inShopnito

inShopnito is an MSA that allows customers and retailers to take advantage of mobile platforms without sacrificing customer's privacy. To achieve this goal, inShopnito relies on the following main components (Figure 1): anonymous credential management, privacy-preserving loyalty and voucher schemes, privacy-preserving recommendations and a secure storage mechanism. Thanks to these mechanisms, inShopnito supports shopping sessions (i.e., purchases, loyalty point and voucher operations) that are anonymous and unlinkable by default; thus, reducing the privacy concerns associated with the data collected by retailers. With inShopnito, customers have better control over their personal data, as identifiable session information is securely stored on their smartphones[1] (instead of the retailer's database). Still, customers can optionally disclose additional information from their local profiles to receive more accurate recommendations and personalized services. Furthermore, inShopnito also provides several benefits to retailers such as support for advanced functionality (e.g., context-based recommendations and family loyalty schemes), improved control and security over loyalty and voucher operations and better integrity guarantees for the anonymous data collected.

A. Threat model

inShopnito's design considers different adversaries. First, *malicious customers* that try to abuse or misuse the application for their benefit. For this purpose, they may attempt to forge, modify, steal or illegally share credentials, loyalty points or vouchers. In addition, they may also try to double-spend loyalty points or vouchers. Second, a *malicious retailer* that tries to break the privacy guarantees offered by inShopnito. For example, a curious but honest retailer may try to discover users' identities by analyzing the data collected during an inShopnito session or by linking different transactions using data mining techniques. However, the retailer follows the protocols set by inShopnito. Third, an *external adversary* that tries to break the security and privacy guarantees offered by inShopnito via passive attacks (e.g., eavesdropping communications), active attacks (e.g., man-in-the-middle attacks) or by getting access to a customer's smartphone.

Furthermore, we assume that the system has been properly initialized and that no linkable or identifiable information is leaked by other operations (e.g., payments) or protocols (e.g., data link or network information).

B. General Description

In this section we describe how the inShopnito app is used by customers during a visit to a retailer's store (e.g., grocery

[1]Secure cloud storage support will be added in the near future.

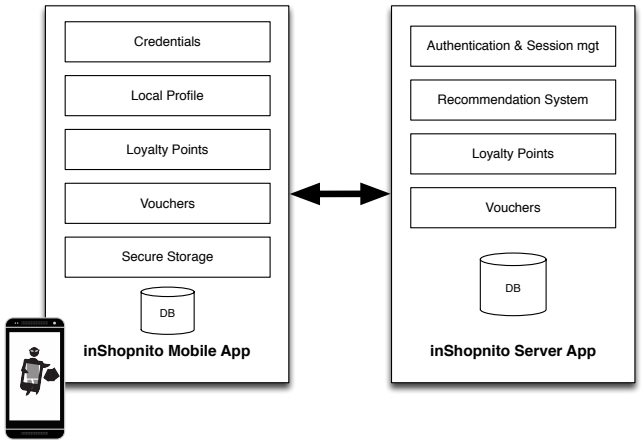

Figure 1. inShopnito's mobile and server components

shop). In particular, we focus on the security and privacy mechanisms used on each step of the shopping session.

Registration: To use the inShopnito app, customers first need to register and obtain an *inShopnito credential*. In our design, the inShopnito credential is an Idemix credential [15] (i.e., anonymous credential) with the minimal attributes required by the retailer. To create their accounts and obtain their credentials, customers can reveal personally identifiable information (e.g., name, address, gender, etc.). inShopnito guarantees that customers' transactions will not be associated with their accounts. The retailer itself or a trusted third-party can act as registration authority. In addition, customers could register remotely over the Internet using the inShopnito app or in person. Furthermore, the phone's secure element is initialized with applets for the secure storage and voucher component[2]. After a successful registration, a customer is issued an inShopnito credential. The credential and its secret are securely stored in the customer's smartphone using inShopnito's secure storage component (Section IV-E). Moreover, credentials can include an expiration date, after which customers will need to renew them.

Starting the app: To start the inShopnito app, customers need to authenticate first. For this purpose, the app prompts customers for a PIN or password (previously set up during app installation) that is used to decrypt and load data from the secure storage (i.e., credential and its secret, loyalty points, vouchers and local profile information). After successful authentication, customers can access all the app information. For instance, loyalty point and voucher balance, previous transactions and information revealed to the retailer can be inspected.

Connecting to the inShopnito server: To start a shopping session, customers need to connect to the retailer's inShopnito server. In our design, we assume that the retailer provides a Wi-Fi network in its facilities to access the server (remote access over a cellular data networks is possible too). The inShopnito app includes a list of the retailer's Wi-Fi access points and servers as well as their corresponding SSL/TLS certificates. Using this information, the inShopnito app automatically connects and authenticates the Wi-Fi access point. Next, the inShopnito app establishes a secure SSL/TLS connection and authenticates the inShopnito server.

Starting an anonymous shopping session: After connecting to the server, the inShopnito client sends a request for a new session to the server. In response, the server sends back its authentication policy and a random nonce. The authentication policy indicates the required and optional attributes the customer should satisfy to start a new session (i.e., customer authentication). For example, the authentication policy may require the customer to prove she owns a valid credential and she is an adult and *optionally* to reveal her shopping preferences and zip code. If the customer accepts the policy, then the inShopnito's credential component (Section IV-A) generates an Idemix zero-knowledge proof that satisfies the policy and sends it to the server. Optionally, the app can remember the customer's acceptance of the current authentication policy. In this way, the customer will only be asked again in the future if the policy changes. If the proof is correctly validated by the server's credential component, then a new shopping cart and a unique session ID are created for the customer. In addition, a copy of the shopping cart is also created in the mobile app. The shopping cart contains the information revealed by the customer in the authentication proof. Still, in most cases this information is not enough to reveal the customer's identity; thus, the session created is anonymous.

Shopping and recommendations: Based on information revealed in the previous step, the customer receives recommendations and discount coupons from the server (Section IV-D). For instance, if the customer only revealed the minimal information required, then she will only receive generic recommendations and coupons. The customer can now proceed to scan items with her smartphone and add them to the inShopnito shopping cart. At any moment before checkout, the customer can add or delete items or modify the amount of a particular item. In addition, the customer can access information about the item (e.g., description, reviews) and about the total amount due. Every time the shopping cart is modified in the mobile app, the server's shopping cart is also updated. Moreover, the server's recommendation component periodically reviews the customer's shopping cart and sends new recommendations and coupons based on the current items in the shopping cart.

Checkout: Before checking out, the customer decides if she wants to redeem loyalty points and vouchers for discounts or other benefits. For example, in our scenario,

[2]As inShopnito aims to be practical, it can also function without secure elements. This, however, disables the voucher component and weakens security guarantees.

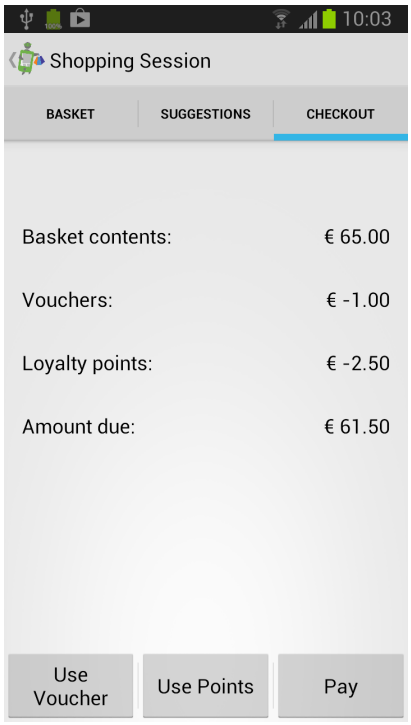

Figure 2. inShopnito's checkout screen. The customer uses one voucher and 250 loyalty points worth €1 and €2.5 respectively.

loyalty points and vouchers are used to provide a direct discount from the total amount due by the customer (1 loyalty point = 1 cent). The loyalty point (Section IV-B) and voucher (Section IV-C) components allow the customer to redeem loyalty points and vouchers in a privacy-preserving way. The customer can select the amount of loyalty points and vouchers to use and review the total discount achieved (Figure III-B). Subsequently, during checkout, the loyalty points and voucher components generate the proofs required to redeem the selected loyalty points and voucher. In addition, if the transaction earns the customer new loyalty points, then the server will send them to the mobile app. Next, a payment operation is executed, which is assumed to be privacy-friendly (e.g., cash or privacy-friendly credit card). Finally, the transaction details are added to local profile component of the inShopnito app.

IV. inShopnito Components

In this section, we present more details of inShopnito's main components (Figure 1).

A. Credentials

The credential component manages the operations associated with inShopnito's credentials in the mobile app and the server. As previously stated, inShopnito's design relies on Idemix credentials [15]; however, support for other anonymous credentials systems is also possible. In the mobile app, the credential component takes care of processing the server authentication policy, creating the corresponding authentication proofs, generating commitments to attributes in the credential and requesting new or updating existent credentials. At the server side, the credential component deals with issuing or updating credentials and validating authentication proofs. Moreover, we use our own development framework [16] to facilitate the integration of Idemix credentials and support future credentials technologies.

B. Loyalty Points

The main goals of the loyalty points component is to allow customers to earn and redeem loyalty points in a privacy-preserving way and to give retailers better control over and security of loyalty point operations. For this purpose, we rely on our own anonymous and unlinkable incentives scheme [17]. By default, inShopnito's loyalty points cannot be associated with a particular transaction or customer identity. Hence, customer privacy is achieved by guaranteeing anonymity and unlinkability while earning and redeeming loyalty points. Still, customers can prove anonymously (via zero-knowledge proofs) that a particular loyalty point is linked to their inShopnito credentials (i.e., the loyalty point was issued to them). This approach allows the retailer to prevent illegal sharing of loyalty points and makes theft more difficult. In addition, the retailer keeps a log of redeemed loyalty points to detect and prevent double-spending. These properties are achieved by using different cryptographic building blocks such as anonymous credentials (i.e., Idemix [15]), zero-knowledge proofs [18], commitment schemes [19] and partially blind signatures [20]. Despite of the heavy cryptographic operations and the use of a mobile platform, our experimental results (Section V) show that our design and implementation are efficient enough for our shopping scenario.

To earn loyalty points, the customer needs to complete a transaction with the retailer. After an interactive protocol, the customer receives a loyalty point that consists of the following values:

$$\mathsf{LoyaltyPoint} = [V, I, Sig(\mathsf{V}, \mathsf{I}, \mathsf{comm}(\mathsf{LS}))]$$

where V is the value of the loyalty point, I is public information that define rules on how the loyalty point should be used (e.g., expiration), $comm(LS)$ is a commitment to a secret value in the customer credential (loyalty secret) and $Sig()$ is a partially blind signature on the previous values generated by the retailer. To ensure a sufficient level of anonymity, the set of possible values and public information combinations should be general and determined in advance (i.e., fixed values). To redeem, the customer sends her loyalty points to the retailer together with a zero-knowledge proof that shows that the customer owns a valid inShopnito credential and that its loyalty secret corresponds with the commitment in the loyalty points to be redeemed.

In addition, the retailer verifies that the loyalty points have not expired, the commitment has not been used before and that the blind signature is correct. For more details about our unlinkable incentives scheme, please refer to our technical report [17].

C. Anonymous Offline Vouchers

Vouchers represent an amount of money issued to customers by the retailer itself (e.g., a refund) or by other parties (e.g., a job benefit). They can also include rules on how they may be spent. The inShopnito voucher component allows customers to receive and spend vouchers with better privacy and security guarantees. In addition, retailers and voucher issuers also benefit from increased control and security. Similarly to loyalty points, our voucher scheme achieves privacy by making voucher issuing and spending operations unlinkable (however, only spending is anonymous). While loyalty points rely on an online system for redeeming and for double-spending prevention, vouchers use an offline approach based on a tamper-resistant secure element (SE) [21]. The voucher component assumes that no data can be extracted from the SE and that the SE executes all tasks as specified in the Voucher Applet (i.e., that it is trustworthy). This ensures that no voucher will ever "exist twice". Vouchers are deleted the moment they are transferred to another SE. As a result, double-spending is prevented. It is clear that this property is broken if a voucher ever leaves the secure environment of the SE in an unencrypted form.

The issuing service has two key pairs: one that it uses to sign vouchers and a second one to prove its identity. Upon registration to the service, each customer or retailer obtains an SE, which has the voucher applet pre-installed. The SE also contains a unique private key and corresponding certificate signed by the issuing service. This private key, generated on the SE, is not known to anybody. The SE uses this key pair to prove that it is indeed certified and that the customer has registered for the service. Furthermore, the SE also contains copies of the two public keys of the issuing service.

A voucher issued to a customer contains the following information:

$$\mathsf{Voucher} = [\mathsf{V}, \mathsf{E}, n, Sig_I(H(\mathsf{V}, \mathsf{E}, \mathsf{P}), n)]$$

where V corresponds to the monetary value of the voucher, E is the expiration date, P define a set of properties (e.g., transferable or not transferable to other customers), n is a random serial number, $H()$ is a cryptographic hash function and Sig_I is a signature on by the issuer. The signature is a partially blinded signature, i.e., the issuer and the customer know the first argument (P), but the issuer has no information on the second argument (n).

The issuing protocol consists of two parts. First we have a mutual authentication stage in which both parties prove their identities. This authentication phase results in a fresh, random shared session key. Once the identities have been verified and the shared key has been established, one or more vouchers can be created for this customer. In this case the recipient proves with a key pair registered to the system that it is an SE. The session key is used to set up a secure and authentic channel between the **SE** and the issuer. This channel is used for the exchange of all messages of the second stage.

For the second stage, we use the partially blinded signature scheme described in [20]. We are using the scheme unaltered with the following conventions: $\mathsf{info} = \mathsf{V}, \mathsf{E}, \mathsf{P}$ and $\mathsf{msg} = n \in_R \mathbb{Z}$. The random number n is generated inside the SE and serves as a unique serial number for the voucher. It is blinded during the signing phase; the issuing service does not know this serial number. The shared parameter info is known to both the customer and the issuing service. In order to ensure sufficient anonymity, the set of possible info values should be limited compared to the number of vouchers. After this second stage, the SE of the customer contains a fresh $\mathsf{Voucher}$. This stage is repeated if more than one voucher is issued.

To transfer one or more vouchers to either another customer or a retailer, we only need to transfer a $\mathsf{Voucher}$ and the corresponding signature to the other party. It is essential during transfer that the receiving party proves that it is in fact a registered SE (to prevent leaking of vouchers outside the system). For privacy, it is essential that the "paying" party does not reveal its identity. Again, first a secure channel is established between the two parties (with the recipient proving its identity). During transfer, the voucher and its signature are encrypted and authenticated using the established session key, preventing an eavesdropper from duplicating it.

D. Local Profile and Recommendation System

A privacy-friendly recommendation system can enhance customer's shopping experience without sacrificing privacy. Many popular recommendation techniques base their suggestions on the behaviour data of all users. inShopnito applies algorithms that can give relevant recommendations to customers using just their own data [22]. The customers manage local profiles on their own device, instead of the retailer gathering vast amounts of data on all customers.

A new shopping basket is created on the server when customers have anonymously authenticated, after which products can be added to this basket. The server includes a recommendation system and a product database containing metadata (e.g., category information, ingredients, etc.) for every item that the store has for sale. The recommendation system suggests products or discount vouchers which have similar metadata as the products that are already in the basket. As more products are added to the basket, the customer's

profile will become more refined and the recommendations are likely to become more accurate.

With inShopnito, the server is unable to link new shopping sessions to any one particular customer. It is not possible to tie any personal information such as age or gender to a certain shopping session, and there is no knowledge of past shopping behavior. Historic consumption behavior is highly useful for recommendation systems, hence the recommendations will initially be of poor quality. This is known as the *cold start problem* [23]. inShopnito therefore offers the customer incentives to voluntarily disclose personal or historical information. Customers can choose to reveal more information than necessary during authentication by revealing additional credential attributes. In addition, inShopnito keeps the customer's shopping history in a local profile on the smartphone itself. This historical information is used to compile a preference profile, which assigns customer preferences to categories. These preferences can be used by the retailer to suggest relevant recommendations.

E. Secure storage

inShopnito's loyalty points and vouchers are an interesting target for attackers. Therefor, inShopnito uses a secure storage component to store the cryptographic keys that are used to decrypt the inShopnito credential, the credential secret and the database secret. The latter is subsequently used to open the encrypted database which contains the customer's local profile. Specific details of this component can be found in [24].

Assuming the device is not rooted, the sandboxes of Windows Phone, iOS and Android provide relatively similar protection against malware. The situation is different with physical attacks [25]. We assume that the customer has not weakened the device security, i.e., a screen lock is configured, ADB is disabled and the device is not rooted. Contrary to Windows Phone 8+ and iOS 4+, file system encryption in Android is neither hardware-backed nor enabled by default[3]. Following the physical attack model, the remaining feasible approach is executing dictionary and brute-force attacks on the encryption passcode [26]. Hardware-backed encryption forces these attacks to take place on the device, making them prohibitively slow. If hardware backing is not present, as is the case on most Android devices, the adversary can leverage offline computing power.

To address this issue, we introduce a secure storage component that relies on the properties of a tamper-resistant secure element, in our case a G&D Mobile Security Card 1.0 [21]. An enrolled app can use its SE to execute trust-sensitive operations (e.g. voucher redemptions) or to access keys in its credential store, which are required for authentication or encryption. The secure element enforces app-specific access through a secret that it shares with the app, due to prior enrollment. Before an app is allowed access to its credential store, the secure element asks for the user's consent by prompting for his passcode. An attempt limit prevents brute-force and dictionary attacks.

[3]Only hardware-backed `KeyChain` storage is provided on supporting devices that run version 4.3 or higher.

V. Performance Evaluation

We have implemented the inShopnito system as an Android application and a Java-based server. The tests were performed on a Samsung Galaxy S3 using Android 4.1.2, while the server is an Ubuntu machine with an Intel Core i7 running at 3.4GHz. Finally, the SE is a G&D Mobile Security Card 1.0 [21].

During application start-up, inShopnito sets up a secure channel to the SE in order to decrypt the inShopnito credential, the credential's secret and the database secret. 15 samples of the application start-up were taken. On average, *7.3 seconds* are required to start the app, of which *3.5 seconds* are taken up by setting up the secure channel with the SE and performing the decryption operations. The remainder is consumed by opening the encrypted database. Note that accessing the SE for decryption and opening the encrypted database are operations that are only performed during application start-up. Hence, we consider this result practical for people to use.

In order to measure the anonymous authentication and session establishment, we have assumed a worst-case scenario were the user has chosen to hide as many attributes in her credential as possible, while still disclosing part of her preference profile. The average for 100 samples is *1.47 seconds* using 2048-bit Idemix keylength and a 5-attribute Idemix credential.

To obtain vouchers, the SE creates a partially blinded signature with the voucher issuer. This SE needs *2.73 seconds* to perform all cryptographic operations for a keylength of 1024-bit. In addition, it needs *1.54 seconds* to verify the integrity of the voucher. Redeeming vouchers only requires a secure connection to be set up between the SEs, which requires under half a second to finish.

We measured the time it takes to earn and redeem sets of loyalty points (see Table I). Note that we do not take the actual value of the points into account, but the total quantity of tokens. The time it takes to earn points is linear to the number of points that is being issued, as one partially blinded signature has to be created for each point. However, the redemption of points is almost independent of the number of loyalty points being redeemed. This is because inShopnito uses only one zero-knowledge proof to assert that all points are linked to the customer's credential, rather than having one proof per point.

Using these results, we can give a conservative estimation of the total checkout time. Assume that a customer redeems one voucher and five separate loyalty points, while in return she earns ten separate loyalty points. This adds up to a total

Table I
TOTAL EARNING AND REDEEMING TIMES OF BATCHES OF 1, 2, 5 AND 10 LOYALTY POINTS. THE TIMES INCLUDE COMPUTATIONS PERFORMED ON CLIENT AND SERVER AS WELL AS THE NETWORK TIME. THE MEAN TIMES FOR 100 SAMPLES ARE LISTED IN MILLISECONDS.

Number of Points	1	2	5	10
Earning (ms)	437	640	1282	2421
Redeeming (ms)	742	754	779	807

time of *3.7 seconds* , which illustrates that inShopnito is indeed practical to use.

VI. DISCUSSION

inShopnito's loyalty points and vouchers rely on signatures to prevent their forgery: only those with a valid signature are accepted by retailers. In addition, it ensures that only the retailer or issuer can create new, valid vouchers and loyalty points, as only they have access to the private signature key. Furthermore, these digital signatures ensure that altering parts of a voucher or loyalty point is detectable. inShopnito denies illegitimate sharing of loyalty points between customers by linking them to a personal inShopnito credential. In order to redeem loyalty points, the customer is required to prove this link using a zero-knowledge proof. Double-spending of loyalty points is prevented by keeping a database containing all spent ones. The retailer records each successfully verified loyalty point. Fortunatly, when the loyalty points contain an expiration date, the retailer can regularly expunge expired points from the database.

Contrary to loyalty points, it can be desirable to hand over vouchers to another person. The voucher's properties determine whether or not it is transferable (cfr. section IV-C). Double voucher spending is prevented through the use of the trusted execution environment and the fact that no voucher will ever leave this environment unprotected. Every time a voucher leaves the SE (for example, when in transit to another SE), the communication link is secured end-to-end, with the end-points being the two communicating SEs. Finally, the issuer keeps a record of vouchers that have been returned. A denial of service attack on inShopnito is possible by executing a large number of authentication requests and session initiations. inShopnito requires customers to first perform an anonymous authentication by proving ownership of their Idemix credential. However, denial of service can be limited by using domain pseudonyms. Every owner of an Idemix credential can generate exactly one valid domain pseudonym per given domain. Hence, retailers can choose a time-frame as domain and enforce the usage of domain pseudonyms by altering the authentication policy. The same domain pseudonym is generated the second time someone tries to authenticate in a given time-frame, which is detectable.

The unlikability of voucher and loyalty point transactions is protected through the use of a partially blinded signature. This enables the issuer to verify *what* it is signing, while it also prevents the issuer from linking issued vouchers and points to redeemed ones at a later stage. Furthermore, inShopnito uses the Idemix credential system to safeguard the customer's anonymity during authentication. This prevents the retailer from linking multiple sessions from the same customer together and thus from building detailed profiles. However, customers have the option to reveal additional data, like credential attributes or data derived from their preference profile. However, if the customer consents to release additional shopping data, an extra measure is taken to protect her privacy. Instead of releasing the precise shopping history down to each individual product, the inShopnito application discloses only what types of products the customer is interested in. The metadata of products that have been purchased in the past is used to construct a preference profile which is then disclosed to the retailer. In addition, customers are able to modify their shopping history and preference profile (e.g., remove a transaction containing a privacy-sensitive product).

A retailer could modify inShopnito's authentication policy so authenticating customers are required to reveal identifiable information to authenticate. That is why *transparency is a key aspect of the inShopnito application*. inShopnito provides a user interface through which the customer can see her shopping history, preference profile and the authentication policy. This way, the customer gains insight into what data has been collected so far, and can take a more informed decision about whether or not to disclose extra information.

The inShopnito application prevents external adversaries monitoring communication channels from learning anything, as all channels are encrypted using SSL/TLS. In addition, man-in-the-middle attacks are prevented, as inShopnito explicitly verifies the certificates used to set up these connections.

inShopnito uses a secure storage component to protect customers against an external adversary who got hold of their smartphone and wants to get access to the vouchers, loyalty points or transactional data. The cryptographic keys used to decrypt the inShopnito credential and the database password are kept in a SE and can only be accessed if the correct passcode is entered. Regarding security, the gap with iOS and Windows Phone is bridged. Not only are offline passcode attacks prevented, the secure element is also blocked as soon as the passcode entry limit is exhausted. This makes data on the device unavailable, in a similar manner as an iOS or Windows Phone wipe. Because of the limited number of attempts, the passcode may be short and non-complex, keeping the usability burden small.

VII. CONCLUSION

In this paper we presented inShopnito, a mobile shopping assistant that offers a rich set of features without sacrificing

the privacy of the user. We accomplished this by integrating multiple advanced components into a single cohesive, usable application. Our privacy-friendly approach to authentication, data storage, loyalty programs, discount vouchers and personalized recommendations provides benefits and protection to both customers and retailers. The proposed solution includes protective measures against forgery and privacy threats originating from malicious customers, overly curious retailers and external adversaries. Our performance evaluation showed that under worst-case circumstances the application still performs adequately.

REFERENCES

[1] Mobestream Media, "Key Ring," http://keyringapp.com/, 2013.

[2] Snapp', "FidMe," http://www.fidme.com/en/home.html, 2013.

[3] PunchTab, "PunchTab," https://www.punchtab.com/, 2013.

[4] Apple, "Passbook," https://www.apple.com/ios/whats-new/#passbook, 2013.

[5] Belgacom Group, "Belgacom and BNP Paribas Fortis jointly launch in-app commerce on smartphones." http://www.belgacom.com/be-en/newsdetail/ND_20130318_Mobile_wallet.page, 2013.

[6] eMarketer, "To Keep Users Happy, Loyalty Programs Must Walk a Fine Line," http://www.emarketer.com/Articles/Print.aspx?R=1009958, 2013.

[7] CASPIAN, "Consumers Against Supermarket Privacy Invasion and Numbering," http://www.nocards.org/, 2004.

[8] V. Pez, "Negative effects of loyalty programs : An empirical investigation on the french mobile phone sector," http://ideas.repec.org/p/ner/dauphi/urnhdl123456789-2379.html, Universit Paris-Dauphine, Tech. Rep., 2007.

[9] S. Worthington and J. Fear, "The hidden side of loyalty card programs," *The Austalian centre for retail studies*, 2009.

[10] I. horn Hann, K. lung Hui, T. S. Lee, and I. P. L. Png, "Consumer privacy and marketing avoidance," in *Equilibrium Analysis: Essays in Honor of*. Cambridge University Press, 2005.

[11] O. Hinz, E. Gerstmeier, O. Tafreschi, M. Enzmann, and M. Schneider, "Customer loyalty programs and privacy concerns," in *Proceedings of BLED 2007*, 2007.

[12] M. Enzmann and M. Schneider, "A privacy-friendly loyalty system for electronic marketplaces," in *IEEE International Conference on e-Technology, e-Commerce and e-Service, 2004. EEE '04*, march 2004, pp. 385 – 393.

[13] P. Marquardt, D. Dagon, and P. Traynor, "Impeding individual user profiling in shopper loyalty programs," in *Proceedings of the Financial Cryptography and Data Security*, ser. FC'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 93–101.

[14] M. Hardt and S. Nath, "Privacy-aware personalization for mobile advertising," in *Proceedings of the 2012 ACM conference on Computer and communications security*, ser. CCS, 2012, pp. 662–673.

[15] IBM Research Security Team, "Specification of the Identity Mixer Cryptographic Library v. 2.3.4," Tech. Rep., 2012.

[16] A. Put, I. Dacosta, and B. De Decker, "Priman: Facilitating the development of privacy-preserving applications," in *IFIP TC-11 SEC 2014 International Conference ICT Systems Security and Privacy Protection*, 2014.

[17] M. Milutinovic, I. Dacosta, A. Put, and B. De Decker, "An efficient and unlinkable incentives scheme," http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW659.abs.html, Internal report, 2014.

[18] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Proceedings of the Annual International Cryptology Conference on Advances in Cryptology*, 1993.

[19] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology CRYPTO 91*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed. Springer Berlin Heidelberg, 1992, vol. 576, pp. 129–140. [Online]. Available: http://dx.doi.org/10.1007/3-540-46766-1_9

[20] M. Abe and T. Okamoto, "Provably secure partially blind signatures," in *Advances in Cryptology CRYPTO 2000*, ser. Lecture Notes in Computer Science, M. Bellare, Ed. Springer Berlin Heidelberg, 2000, vol. 1880, pp. 271–286.

[21] Giesecke & Devrient, "Mobile Security Card SE 1.0," http://www.gd-sfs.com/the-mobile-security-card/mobile-security-card-se-1-0/.

[22] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper." Citeseer, 1999.

[23] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proceedings of the ACM conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2002, pp. 253–260.

[24] F. Boukayoua, J. Lapon, B. D. Decker, and V. Naessens, "Improving secure storage of data in android," KU Leuven, https://lirias.kuleuven.be/handle/123456789/446974, Internal Report, March 2014.

[25] T. Vidas, D. Votipka, and N. Christin, "All your droid are belong to us: A survey of current android attacks," in *Proceedings of the 5th USENIX Conference on Offensive Technologies*, ser. WOOT'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 10–10.

[26] P. Teufl, T. Zefferer, and C. Stromberger, "Mobile device encryption systems," in *Proceedings of the 28th IFIP TC-11 SEC 2013 International Information Security and Privacy Conference*, Auckland, New Zealand, Jul. 2013, p. 15.