A Privacy-Protecting Multi-Coupon Scheme with Stronger Protection Against Splitting

Liqun Chen[1], Alberto N. Escalante B.[2], Hans Löhr[2],
Mark Manulis[2], and Ahmad-Reza Sadeghi[2]

[1] HP Laboratories
liqun.chen@hp.com
[2] Horst Görtz Institute, Ruhr-University of Bochum, Germany
{eban,hloehr,sadeghi}@crypto.rub.de, mark.manulis@rub.de

Abstract. A multi-coupon (*MC*) represents a collection of k coupons that a user can redeem to a vendor in exchange for some goods or services. Nguyen (FC 2006), deepening the ideas of Chen et al. (FC 2005), introduced an unforgeable privacy-protecting *MC* system with constant complexity for issuing and redemption of *MC*s, that discourages sharing of coupons through a property called *weak unsplittability*, where sharing of a single coupon implies sharing of the whole multi-coupon (all-or-nothing sharing). Both schemes still lack some features required by many applications in practice, and also stronger forms of unsplittability are desirable. In this paper, we propose a new security model for *MC* systems with stronger definitions, followed by a concrete realization where single coupons within a *MC* may represent different goods or services, have independent validity periods, and must be redeemed sequentially ensuring a stronger version of unsplittability compared to all-or-nothing sharing. The complexity of the proposed scheme is linear in k for the generation of multi-coupons and constant for each redeemed single coupon.

Keywords: Coupon, privacy, unsplittability, unlinkability, loyalty.

1 Introduction

Paper-based coupon schemes are successfully used by enterprises for various marketing purposes like providing discounts, increasing sales within a period of time (via coupons with some specified validity period), setting up prepayment models, attracting new customers, and establishing long-term relationships (loyalty) with them. From an abstract point of view, a *coupon* is some information that gives a customer the right to claim a good or service from a vendor.

The procedure in which a vendor provides a customer with a new coupon is called *issue*. The procedure in which the customer pays using the obtained coupon is called *redeem*. Here, the vendor verifies that the coupon is valid and authentic, and provides the customer with the specified good or service. Coupons can be used only once. In the following, we denote by *object* the good or service implied by a coupon. Any item that can be bought may become an object in

S. Dietrich and R. Dhamija (Eds.): FC 2007 and USEC 2007, LNCS 4886, pp. 29–44, 2007.

practice, e.g. cloths, songs, books, videos, medicines, tickets, and even immaterial services: discounts, access to computer resources or facilities, etc.

In contrast to widely used paper-based coupon schemes, *electronic coupons* (*e-coupons*) have gained acceptance relatively slowly [14], and are still waiting for their breakthrough. One of the reasons for this development is insufficient security of available schemes. A *multi-coupon* (*MC*) [7,11] denotes a collection of e-coupons that is handled as a single unit.

In this paper we consider a *multi-coupon scheme* (*MCS*) that protects the privacy of the customers, and encourages loyalty of clients by providing *unsplittability* [7], i.e., two users cannot redeem coupons from the same *MC* separately and independently. Consider prepaid-goods, where a vendor, hoping for a long-term client relation, sells many goods at once at a cheaper price compared to that of separately sold goods. In this case sharing would allow a group of users to buy a single *MC*, and obtain goods at a subsided price, but without giving loyalty in return. We focus on a basic *MC* framework where the only involved parties are many customers (users) and a single vendor. Note that other frameworks are imaginable, e.g., with several cooperating vendors.

From the security point of view, threats in *MC* systems are different from those in paper-based coupon systems. First, it is very easy to create a perfect (digital) copy of an electronic *MC*, whereas copying a paper-based booklet requires much higher effort. Second, when dealing with a *MCS* we must also consider attacks in which different users collude and attempt to cheat the vendor. Moreover, in the digital world privacy and anonymity of customers becomes more important since the vendor may try to infer and store additional information about them including purchase habits, gender, age, etc. This would harm privacy and allow client profiling and price discrimination [13], e.g., different customers are offered the same goods by the same vendor, but at different prices.

1.1 Desired Security Properties

We focus on *unforgeability*, *unlinkability*, and *unsplittability* because, as pointed out in [6,7,11], these are the essential properties of a *MCS*.

Unforgeability. There is an intrinsic monetary value associated to any coupon, explicitly or implicitly. Therefore, vendors want their multi-coupons to be *unforgeable*, in the sense that no coalition of users should be able to redeem more coupons than it has been rightfully allowed.

Unlinkability. It must be infeasible for a vendor to link a redeem procedure for a customer to the corresponding issue procedure, or to link two different redeem procedures with the same customer. This implies anonymity of customers.

Unsplittability. *Weak unsplittability* (WU) [7], also known as *all-or-nothing sharing*, intuitively, requires that whenever a user intends to share a single coupon with a second user, she has to provide her with *all* the secret information related to the involved *MC*. This, however, would make possible the complete redemption of the *MC* by the second user. Thus, in case that both users do not trust each other, WU discourages sharing.

A stronger version, called *(ordinary) unsplittability*, requires that it is infeasible for an adversary to produce more *autonomous* redemption algorithms than the number of multi-coupons he has rightfully obtained, where by autonomous we mean that such algorithms do not share any information gained during the redemption. In other words, if a user gives a single coupon to another user, then that second user has to send back some information to the first user after redeeming; otherwise the first user cannot spend further coupons from that multi-coupon. Hence, sharing is more cumbersome with this stronger version of unsplittability than with weak unsplittability because it requires a trust relationship and additional interaction between the users.

Contribution and Organization. We start in Section 2 with the description of related work on multi-coupon schemes, and give a brief overview of our construction in Section 3. In Section 4, we define the syntax and correctness of a *MCS*, and propose a more precise security model for *MCS*s that includes a stronger form of unsplittability without relying on all-or-nothing sharing. Thereafter, we propose in Section 5 a construction of a privacy-protecting *MCS* which satisfies our stronger requirements and provides additional features for practical applications, e.g., different objects for individual coupons within one multi-coupon and validity periods thereof. Redeem complexity (both computation and communication) is constant w.r.t. the size k of the multi-coupon (i.e., the number of coupons it contains), and complexity of the protocol for issuing multi-coupons is linear in k, which is the best we can get when each coupon has individual attributes. Additionally, we prove the security of our scheme w.r.t. the proposed security model. Finally, we provide in Section 6 some insights into possible improvements and future work.

2 Related Work

Syverson et al. [17] introduced the concept of unsplittability in the context of unlinkable serial transactions to discourage sharing, and suggested an extension of their scheme to implement coupon books. Later, Chen et al. [7] described the properties that a privacy-protecting multi-coupon system must provide, justified the use of unsplittability over other means to discourage sharing (e.g., hiding credit card numbers in the multi-coupons), and proposed an unforgeable, unlinkable, and weakly unsplittable scheme. However, their construction is less practical because of an expensive proof of knowledge used in the redemption, whose complexity is linear in k (i.e., the number of coupons in the multi-coupon).

More recently, Nguyen [11] addressed some disadvantages of [7], and defined a security model for *MCS*s, followed by an efficient construction based on a verifiable pseudorandom function and bilinear groups. Its issue and redeem complexity is constant w.r.t. k, it offers the same security properties as in [7], and adds a new feature to *revoke* multi-coupons. It is arguable whether revocation is indeed necessary for a *MCS*, since in real life it is unusual that a vendor revokes issued coupon booklets, and this operation might be costly.

One drawback of both above mentioned schemes is that every issued multi-coupon must contain the same number of coupons, i.e., k is a system parameter fixed for all multi-coupons. This limitation, as pointed out in [11], can be overcome in both schemes by extending the issue protocol. However, this extension is impractical, i.e., for [11] a term $k - m'$ is added to the complexity of the issue protocol, where m' $(0 < m' < k)$ is the number of issued single coupons. Another drawback of these schemes is that there is no concept of coupon's object (or coupon's type [6]). Hence, all coupons are valid for the same purpose.

As previously explained in [7,11], most related schemes (e.g., e-cash, digital credentials) cannot be employed as privacy-protecting unsplittable *MCS*s because they have different usage patterns [15,1], are inefficient in this setup [12], or lack at least one of the required properties [3], in particular unsplittability. Some e-cash systems can be used as unlinkable or at least anonymous *MCS*s (e.g. [4,6]). However, they are (unintentionally) at most weakly unsplittable.

3 Short Overview of Our Construction

In our scheme, each single *redeemable* coupon $(id, ob, sq, \sigma, \sigma')$ is specified by a coupon identifier id, a coupon sequence number sq, a coupon's object ob (i.e., the good or service represented by the coupon[1]), a signature σ on the tuple (id, ob, sq), and a signature σ' on sq. A coupon is not redeemable if it lacks σ'.

A multi-coupon M of size k is a list of k single coupons with consecutive sequence numbers, where at least the first coupon must be redeemable. In the issue protocol, the user obtains a multi-coupon where the coupon identifiers are kept private by the user, and all other attributes are known to both user and vendor. After the issue procedure, only the first coupon is redeemable, but every coupon has a valid signature σ_i, for $0 \leq i < k$. During the redemption of the i-th single coupon with sequence number sq_i, the user obtains a signature σ'_{i+1} on the sequence number $sq_i + 1$, and hence the next coupon in the list becomes redeemable. In order to redeem a coupon, the user must prove that the coupon has never been used before (by disclosing id), and that it is indeed redeemable (by proving that σ is a valid signature on id, ob and sq, and that σ' is a valid signature on sq).

Informally, the vendor's knowledge about elements of a single coupon depends on the actual procedure, i.e., id is hidden during the issue protocol, and disclosed to the vendor during redemption; sq, σ, σ' are known to the vendor during issuing, but hidden during the redeem protocol; ob is known to the vendor during both the issue and the redeem protocols.

Our scheme utilizes a digital signature scheme with efficient protocols that allows to obtain a signature on a (partially) blinded tuple (i.e., some elements of the tuple are disclosed, while others are only committed to), and to prove the knowledge of a signature on a (partially) blinded tuple without disclosing any useful information, other than the fact that the signature is valid.

[1] The vendor must publish an official coding of coupon's objects as integers.

4 Security Framework for Multi-coupon Schemes

Notation. For a finite set $\mathcal{S}$, $s \in_R \mathcal{S}$ denotes the assignment to the variable s of an element uniformly sampled from $\mathcal{S}$. Let A be a probabilistic algorithm. By $\mathrm{out}_A \leftarrow A(\mathrm{in}_A)$ we denote that the variable out_A is assigned the output of A's execution on input in_A. We denote by $(A(\mathrm{in}_A), B(\mathrm{in}_B))$ a pair of interactive algorithms with private inputs in_A and in_B, respectively, and write $(\mathrm{out}_A, \mathrm{out}_B) \leftarrow (A(\mathrm{in}_A), B(\mathrm{in}_B))$ to denote the assignment of A's and B's private outputs after their interaction to the variables out_A and out_B, respectively.

4.1 General Multi-coupon Schemes

We consider a basic framework where the participants are a single vendor $\mathcal{V}$ and a collection of users $\mathcal{U}_i$. The following definition is general in that it does not account for specific coupon features such as revocation, coupon objects, or validity periods. We will refer to any particular user simply by $\mathcal{U}$.

Definition 1 (Multi-Coupon Scheme). *A* multi-coupon scheme *(MCS) consists of a set of protocols: $\{$`Setup`, `Issue`, and `Redeem`$\}$, which are specified by the following algorithms.*

***Setup* algorithm**. *$(PK, SK) \leftarrow \mathtt{Setup}(1^\kappa)$ is the initialization algorithm executed by the vendor once to generate one instance of the multi-coupon scheme. It takes as input the security parameter κ, and outputs a public key PK (which from now on we assume to include the security parameter κ coded in unary, and a system parameter $k_{\max}$ representing the maximum allowed number of coupons per MC), and a secret key SK (which might include PK).*

***Issue* protocol**. *In order to obtain a MC with k coupons, $\mathcal{U}$ performs the following protocol with $\mathcal{V}$: $((res_u, M), res_v) \leftarrow (\mathtt{Issue}_u(k, PK), \mathtt{Issue}_v(k, SK))$, where, from now on, the subindices u and v denote user and vendor algorithms, respectively. The output flags res_u, $res_v \in \{acc, rej\}$ indicate success or failure according to the user or vendor, resp. $\mathtt{Issue}_u$ outputs the flag res_u and a multi-coupon M, whereas $\mathtt{Issue}_v$ only outputs the flag res_v.*

***Redeem* protocol**. *After $\mathcal{U}$ has obtained the multi-coupon M she redeems it to $\mathcal{V}$ by performing the protocol $((res_u, M'), (res_v, \varsigma')) \leftarrow (\mathtt{Redeem}_u(M, PK), \mathtt{Redeem}_v(\varsigma, SK))$. $\mathtt{Redeem}_u$ outputs an updated multi-coupon M', and a flag res_u just like in issue, and $\mathtt{Redeem}_v$ outputs a new vendor's internal state ς', which is initially set to the empty string, and a flag res_v.*

The correctness requirement states that an honest user who obtains a *MC* from a fresh honest vendor must be able to redeem all the coupons it contains.

Definition 2 (Correctness). *A multi-coupon scheme is* correct *if the following experiment returns true with overwhelming probability (for any $k \in [1, k_{\max}]$), where $resIs$ and $resRe$ are the output flags of the issue and redeem algorithms, respectively, and ς_i is the vendor's state, which is updated after each redemption.*

```
$(PK, SK) \leftarrow \texttt{Setup}(1^\kappa);\ \varsigma_1 \leftarrow \varepsilon;$
$((resIs_u, M_1), resIs_v) \leftarrow (\texttt{Issue}_u(k, PK), \texttt{Issue}_v(k, SK));$
for $i = 1$ to $k$ do:
    $((resRe_u^i, M_{i+1}), (resRe_v^i, \varsigma_{i+1})) \leftarrow (\texttt{Redeem}_u(M_i, PK), \texttt{Redeem}_v(\varsigma_i, SK));$
if $(resIs_u, resIs_v, resRe_u^1, resRe_v^1, \ldots, resRe_u^k, resRe_v^k) = (acc, \ldots, acc)$
    return $true$; else return $false$;
```

4.2 Adversarial Model and Security Requirements

In this section we present a solid security framework that covers a wide range of adversarial actions. We begin by defining the queries available to the adversary, and then we define the security requirements.

An adversary is a p.p.t. algorithm $\mathcal{A}$, which can play the role of, either, a vendor and a group of users, or only of a group of users. $\mathcal{A}$ can interact with the other participants through a set of queries, which cannot be interleaved.[2] Wlog we let the adversary be specified by a sequence of algorithms (e.g. $\mathcal{A} := (A_1, A_2, A_3)$). Honest parties are assumed to communicate over secure channels.

Depending on the degree of independence from the adversary, we consider two types of users: *scheduled* and *corrupted users*. Users belonging to the set of scheduled users ($\mathcal{SU}$) execute honest algorithms if requested by the adversary, but remain honest otherwise. The adversary has full control over the corrupted users, grouped in the set $\mathcal{CU}$, and is provided with their previous protocol views. Additionally, the adversary might act as a group of malicious users.

Similar to [9], we allow the adversary to interact with the system through a set of queries handled by an *interface*, which partially simulates the *MCS*, executes protocols with the adversary, and records certain user's or vendor's activities. The queries available to an adversary differ depending on whether he is playing the vendor's role or only a user coalition. We distinguish between two types of interfaces. The first interface (I_1) is employed to model a *MCS* facing a collusion of users, and is used to define unforgeability, and unsplittability. The second interface (I_2) models a *MCS* controlled by a malicious vendor, and is only employed to define unlinkability.

Interface 1 (I_1). In this case the adversary plays a collusion of users, and the interface plays the vendor and the honest users. I_1 maintains the vendor's state ς, and some counters, which are updated in each query: χ_M: number of non-empty multi-coupons rightfully provided to the adversary, χ_C^x: number of available (used and unused) coupons given to x, and χ_R^x: number of coupons redeemed by x, where x denotes one of the participants, and can be either A to denote the adversary, or some arbitrary string $\mathcal{U}$ to denote a particular user. Now we present the queries and the actions performed by the interface.

I_1.GetPK. Returns the vendor's PK to the adversary.

$I_1.\texttt{Issue}_v(k)$. If $k < 1$ or $k > k_{\max}$ the interface aborts (halts and returns *rej*), otherwise it simulates the $\texttt{Issue}_v$ algorithm playing the vendor, and interacts

[2] This reflects the properties of existing schemes, and simplifies the construction.

with the adversary, who plays the user. The counters are updated as follows: χ_M++, χ_C^A+=k (where ++ and +=k denote increment by 1 and k, resp.).

I_1.$\texttt{Issue}_u(\mathcal{U}, k)$**.** If $k < 1$, $k > k_{\max}$, $\mathcal{U} \in \mathcal{SU}$, or $\mathcal{U} \in \mathcal{CU}$, then the interface aborts, otherwise it simulates a protocol run between an honest user $\mathcal{U}$ and the vendor. $\mathcal{U}$ is an arbitrary value specified by the adversary to the interface, which allows the adversary to refer to precisely the same user later on. The user's view of the protocol is stored in a *transcript*, and the variables are updated: $\mathcal{SU} \leftarrow \mathcal{SU} \cup \{\mathcal{U}\}$, $\chi_C^{\mathcal{U}} \leftarrow k$, $\chi_R^{\mathcal{U}} \leftarrow 0$. In our security model every existing user has exactly one multi-coupon: a real world honest user with m multi-coupons can be simulated by m users, each one having a single multi-coupon.

I_1.$\texttt{Redeem}_v$**.** The interface performs the $\texttt{Redeem}_v$ algorithm, enabling the adversary to redeem one of his coupons. If the interaction is successful ($res_v = acc$) the counters are updated as follows: χ_R^A++, $\chi_M \leftarrow \min(\chi_M, \chi_C^A - \chi_R^A)$. (An adversary with at most $\chi_C^A - \chi_R^A$ unused coupons is not allowed to have more than $\chi_C^A - \chi_R^A$ non-empty multi-coupons.) These counters are important for the unforgeability and unsplittability requirements.

I_1.$\texttt{Redeem}_u(\mathcal{U})$**.** The interface simulates a $\texttt{Redeem}$ protocol run between the honest user $\mathcal{U}$ and the vendor (both algorithms $\texttt{Redeem}_u$ and $\texttt{Redeem}_v$ are simulated). If $res_v = acc$ the interface stores the user's view in the *transcript*, and sets $\chi_R^{\mathcal{U}}$++. The only information returned to the adversary is res_v.

I_1.$\texttt{Corrupt}(\mathcal{U})$**.** The interface first verifies that $\mathcal{U} \in \mathcal{SU}$, otherwise it aborts. Then it sets $\mathcal{SU} \leftarrow \mathcal{SU} \backslash \{\mathcal{U}\}, \mathcal{CU} \leftarrow \mathcal{CU} \cup \{\mathcal{U}\}$, and finally it gives to the adversary the user's previous protocol views, which are extracted from the *transcript*. The counters are updated: χ_C^A+=$\chi_C^{\mathcal{U}}$, χ_R^A+=$\chi_R^{\mathcal{U}}$, and if $\chi_C^{\mathcal{U}} > \chi_R^{\mathcal{U}}$, then χ_M+=1.

Interface 2 (I_2). This interface is capable of simulating a collection of honest users scheduled by the adversary, who plays the vendor. Again we use $\mathcal{SU}$ and $\mathcal{CU}$ to denote sets of scheduled and corrupted users resp., and the counters χ_C^x and χ_R^x with the same meaning as in I_1. The following queries are provided:

I_2.$\texttt{GetPK-SK}$. The interface gives the pair (PK, SK) to the adversary.

I_2.$\texttt{Issue}_u(\mathcal{U}, k)$. If $k \in [1, k_{\max}]$ and $\mathcal{U} \notin \mathcal{SU} \cup \mathcal{CU}$ the interface executes the $\texttt{Issue}_u$ algorithm (otherwise it aborts). Then, interface sets $\mathcal{SU} \leftarrow \mathcal{SU} \cup \{\mathcal{U}\}$, $\chi_C^{\mathcal{U}} \leftarrow k$, $\chi_R^{\mathcal{U}} \leftarrow 0$, and appends its protocol view to the *transcript*.

I_2.$\texttt{Redeem}_u(\mathcal{U})$. If $\mathcal{U} \notin \mathcal{SU}$ or $\chi_C^{\mathcal{U}} = \chi_R^{\mathcal{U}}$, then the interface aborts (the second condition prevents the interface from trying to overuse a multi-coupon). Then it executes the $\texttt{Redeem}_u$ algorithm simulating the honest user $\mathcal{U}$. The vendor stores the user's view in the *transcript*, and sets $\chi_R^{\mathcal{U}}$++.

I_2.$\texttt{Corrupt}(\mathcal{U})$. This query is handled exactly as in I_1.

4.3 Unforgeability

Informally, unforgeability means that no group of users (controlled by $\mathcal{A}$), with χ_C^A coupons in total (comprised in, say, m multi-coupons), should be able to redeem $\chi_R^A > \chi_C^A$ coupons. More formally, this property is defined as follows.

```
$ForgeGame(\mathcal{A}, \kappa)$:
    $(PK, SK) \leftarrow \texttt{Setup}(1^\kappa)$;
    $\sigma \leftarrow A_1^{I_1}(1^\kappa)$;
    if $(\chi_C^A \neq \chi_R^A)$ then return $unbroken$;
    $(res_A, res_v) \leftarrow (A_2(\sigma), I_1.\texttt{Redeem}_v)$;
    if $(res_v = acc)$ then return $broken$;
        else return $unbroken$;
```

```
$SplitGame(\mathcal{A}, \kappa)$:
    $(PK, SK) \leftarrow \texttt{Setup}(1^\kappa)$;
    $(\sigma_0, \ldots, \sigma_{\chi_M}) \leftarrow A_1^{I_1}(1^\kappa)$
    for $i = 0$ to $\chi_M$ do:
        $(res_A^i, res_v^i) \leftarrow (A_2(\sigma_i), I_1.\texttt{Redeem}_v)$;
    if $(res_v^0 = acc \wedge \ldots \wedge res_v^{\chi_M} = acc)$ then
        return $broken$; else return $unbroken$;
```

Fig. 1. Forgeability and Splittability Games

Definition 3 (Unforgeable *MCS*). *A multi-coupon scheme is* unforgeable *if there is no p.p.t adversary $\mathcal{A} := (A_1, A_2)$ that can win the forgeability game in Fig. 1 (ForgeGame$(\mathcal{A}, \kappa)$ = broken) with non-negligible probability (in κ).*

An adversary $\mathcal{A}$ first interacts with the interface I_1 (i.e., queries $\texttt{GetPK}$, $\texttt{Issue}_v(\cdot)$, $\texttt{Issue}_u(\cdot,\cdot)$, $\texttt{Redeem}_v$, $\texttt{Redeem}_u(\cdot)$, and $\texttt{Corrupt}(\cdot)$). $\mathcal{A}$ wins if he is able to redeem an additional coupon after having redeemed the *same* number of coupons he has rightfully obtained. Note that any adversary $\mathcal{A}'$ who achieves $\chi_R^A > \chi_C^A$, can be transformed into an adversary $\mathcal{A}$, who wins the *ForgeGame* at the expense of at most a polynomial factor in the success probability.

4.4 Unsplittability

Informally, a *MCS* is *unsplittable* if it is infeasible for an adversary $\mathcal{A}$ rightfully holding at most χ_M non-empty *MC*s to generate $\chi_M + 1$ *shares* $\sigma_0, \ldots, \sigma_{\chi_M}$, which can be used each to autonomously redeem at least one coupon. This must hold, even though $\mathcal{A}$ might have $\chi_C^A - \chi_R^A \geq \chi_M$ unused coupons.

Definition 4 (Unsplittability). *A multi-coupon scheme is* unsplittable *if there is no p.p.t. adversary $\mathcal{A} := (A_1, A_2)$ capable of winning the splittability game in Fig. 1 (SplitGame$(\mathcal{A}, \kappa)$ = broken) with non-negligible probability (in κ).*

In the splittability game the adversary first interacts with the interface I_1, and outputs $\chi_M + 1$ indexed states (shares) $\sigma_0, \ldots, \sigma_{\chi_M}$. Then he sequentially executes $\chi_M + 1$ redemption algorithms $A_2(\sigma_i)$, for $0 \leq i \leq \chi_M$. The adversary wins if each one of the $\chi_M + 1$ redemption algorithms succeeds.

We remark that, inside the "for loop" in Fig. 1, $A_2(\sigma_i)$ does not depend on the information obtained in the execution of $A_2(\sigma_j)$ with $i \neq j$. The adversary's only input is a state σ_i (for some i); this ensures the autonomous redemption. In contrast, the interface I_1 implicitly updates the vendor's state.

4.5 Unlinkability

Informally speaking, unlinkability means that an adversary playing the role of the vendor cannot recognize (significantly better than by a random guess) which honest user redeems a coupon when such a user is randomly selected from a pair of users of his choice (equivalently with *MC*s instead of users).

In [6] a simple definition of unlinkability is proposed. However, the adversary cannot further interact with the users after the challenge took place.

The number of unused coupons left in the selected pair of *MC*s can be easily used by the adversary to link the protocols. This problem is (almost) solved in [11] by hiding the number of unused coupons of the pair of challenged *MC*s from the adversary. However, this is done (in part) by requiring that none of the challenged *MC*s is ever emptied, hence the adversary is unrealistically prevented from using the last coupons within the challenged *MC*s.

Definition 5 (Unlinkability). *A multi-coupon scheme is* unlinkable *if there is no p.p.t. adversary* $\mathcal{A} := (A_1, A_2, A_3)$ *with non-negligible linkability advantage, which is defined as:* $Adv^{link}(\mathcal{A}, \kappa) = Pr[LinkGame(\mathcal{A}, \kappa) = broken] - 1/2$.

For the linkability game, the adversary $\mathcal{A}$ first interacts with the interface I_2 (queries $\mathtt{GetPK\text{-}SK}$, $\mathtt{Issue}_u(\cdot,\cdot)$, $\mathtt{Redeem}_u(\cdot)$, and $\mathtt{Corrupt}(\cdot)$), and outputs the user identities $\mathcal{U}_0$ and $\mathcal{U}_1$, of two scheduled users that have at least one unused coupon left (i.e. $\chi_C^x > \chi_R^x$, for $x \in \{\mathcal{U}_0, \mathcal{U}_1\}$). Then, b is randomly selected from $\{0, 1\}$, and the redemption algorithm $\mathtt{Redeem}_u(\mathcal{U}_b)$ is executed with $\mathcal{A}$. Afterwards, $\mathcal{A}$ is given a set of queries $I_2(m_0, m_1, \mathcal{U}_0, \mathcal{U}_1)$, similar to those of I_2, except that the users $\mathcal{U}_0$ and $\mathcal{U}_1$ cannot be corrupted, and at most m_0 $\mathtt{Redeem}_u(\cdot)$ queries can be made for the user $\mathcal{U}_0$ and m_1 queries for $\mathcal{U}_1$, where m_0 (resp. m_1) is the number of unredeemed available coupons minus one held by user $\mathcal{U}_0$ (resp. $\mathcal{U}_1$) before A_2 redeems. This hides the number of unused coupons from $\mathcal{A}$, thus avoiding the problem mentioned above. Finally, $\mathcal{A}$ outputs d. If $d = b$ the adversary won the game, otherwise he lost.

```
$LinkGame(A_1, A_2, A_3, \kappa)$:
  $(PK, SK) \leftarrow \mathtt{Setup}(1^\kappa)$;
  $(\mathcal{U}_0, \mathcal{U}_1, \varsigma) \leftarrow A_1^{I_2}(1^\kappa)$;
  if not $(\mathcal{U}_0 \in \mathcal{SU} \wedge \mathcal{U}_1 \in \mathcal{SU} \wedge \chi_C^{\mathcal{U}_0} > \chi_R^{\mathcal{U}_0} \wedge \chi_C^{\mathcal{U}_1} > \chi_R^{\mathcal{U}_1})$
    then return $unbroken$;
  $b \leftarrow \{0, 1\}$;  $m_0 \leftarrow \chi_C^{\mathcal{U}_0} - \chi_R^{\mathcal{U}_0} - 1$;  $m_1 \leftarrow \chi_C^{\mathcal{U}_1} - \chi_R^{\mathcal{U}_1} - 1$;
  $(res_{\mathcal{U}_b}, \varsigma) \leftarrow (I_2.\mathtt{Redeem}_u(\mathcal{U}_b), A_2(\varsigma))$;
  $d \leftarrow A_3^{I_2(m_0, m_1, \mathcal{U}_0, \mathcal{U}_1)}(\varsigma)$;
  if $(res_{\mathcal{U}_b} = acc \wedge d = b)$ then return $broken$; else return $unbroken$;
```

Theorem 1. *Unsplittability is strictly stronger than unforgeability.*

Proof (Sketch). ($\Rightarrow$) The condition $\chi_C^A = \chi_R^A$ in the forgeability game implies $\chi_M \leq 0$. Therefore, an adversary $\mathcal{A}$ against *ForgeGame* is also an adversary against *SplitGame* with at least the same success probability. E.g., if $\chi_M = 0$, then $\mathcal{A}$ "splits zero multi-coupons into one". For the other direction ($\not\Leftarrow$) consider the schemes proposed in [7,11] which are unforgeable but not unsplittable.

5 Our Multi-coupon Scheme

We propose the first unsplittable *MCS* where each coupon has an individual object, and coupons belonging to the same *MC* must be redeemed in certain linear order, which is fixed during the issue procedure. The scheme can be easily extended with validity periods and arbitrary attributes for each coupon. In contrast to previous proposals [7], the number of coupons contained in a multi-coupon is not fixed, but is upper-bounded by $k_{\max}$. Therefore, no inefficient step is required for issuing a fraction of the maximum number of coupons [11]. This is useful, for instance, to implement a personalized electronic discount booklet, where variable discounts are offered in certain order.

5.1 Notation and Building Blocks

Commitment Scheme (CS). We use the integer tuple CS from [10], based on the scheme in [8], with a tuple $(g_1, \ldots, g_k, n)$ of k bases $g_i \in \mathrm{QR}_n$ (quadratic residues modulo n), for $1 \leq i \leq k$, and a special RSA modulus n as a public key. A commitment to $(x_1, \ldots, x_{k-1})$ has the form $C_x = g_1^{x_1} \cdots g_k^{x_k}$, where x_k is a value randomly chosen from an appropriate interval.

Proofs of Knowledge (PoK). We use a number of honest-verifier statistical zero-knowledge PoK. By $\mathtt{PoK}\{(\tilde{x}_1, \ldots, \tilde{x}_n) : R(\tilde{x}_1, \ldots, \tilde{x}_n)\}$ we denote an interactive PoK, where a prover proves to a verifier that she knows a witness $(\tilde{x}_1, \ldots, \tilde{x}_n)$ (which we always denote with tilded variables) such that the relation R holds, and the verifier does not gain any useful information beyond this assumption.

Proof of Equality of Representations. $\mathcal{P}$ proves that she is able to open two commitments C_1 and C_2 (for two possibly different instances of the commitment scheme), such that certain components of the openings are equal. For example, we write $\mathtt{PoKEqRep}\{(\tilde{x}, \tilde{r}_x, \tilde{y}, \tilde{r}_y) : C_1 = g_1^{\tilde{x}} g_2^{\tilde{r}_x} \wedge C_2 = \hat{g}_1^{\tilde{y}} \hat{g}_2^{\tilde{r}_y} \wedge \tilde{x} = \tilde{y}\}$.

Camenisch Lysyanskaya signature scheme (*CLS*). The *CLS* [5] is a simple signature scheme with efficient protocols based on the *strong RSA assumption*. The following description is done in the context of our scheme.

CLS.$\mathtt{Setup}(1^\kappa)$. The signer $\mathcal{S}$ generates a special RSA modulus $n = pq$, such that n has size $\ell_n := 2\kappa$, where κ is a security parameter. Then he chooses numbers $a, b \in_R \mathrm{QR}_n$ called bases, and a constant $c \in_R \mathrm{QR}_n$. The public key CLS_{PK} is (a, b, c, n), and the secret key CLS_{SK} is the prime number p.

CLS.$\mathtt{Sign}(x, CLS_{SK})$. To sign a message $x \in [0, 2^{\ell_m})$, the signer chooses a random prime e of size exactly $\ell_e := \ell_m + 2$, a random number s of size at most $\ell_s := \ell_n + \ell_m + \ell$, where ℓ is another security parameter, $\mathcal{S}$ computes $v \leftarrow (a^x b^s c)^{e^{-1}} \pmod{n}$, and outputs (e, s, v).

CLS.$\mathtt{Verify}(x, \sigma, CLS_{PK})$. For $(e, s, v) := \sigma$, the algorithm tests whether $v^e \equiv a^x b^s c \pmod{n}$, $x \in [0, 2^{\ell_m})$, $s \in [0, 2^{\ell_s})$, e is exactly ℓ_e bits long, and outputs *true* or *false* accordingly.

The signature allows the following useful protocols:

Signature on a committed value and PoK of this signature [5]. Signature generation is a protocol from [5, Fig. 1] between a user $\mathcal{U}$ and a signer $\mathcal{S}$, who knows the secret key CLS_{SK}. (Let $CLS_{PK} := (a, b, c, n)$ be the corresponding public key.) The common input to $\mathcal{U}$ and $\mathcal{S}$ is a commitment C_x, for which $\mathcal{U}$ supposedly knows an opening $(x, r_x) : C_x = a^x b^{r_x}$. At the end of the protocol $\mathcal{U}$

obtains a signature $\sigma := (e, s, v)$ on x, while x is statistically hidden from $\mathcal{S}$. We denote this protocol as: $\sigma \leftarrow \mathtt{SigOnCommit}\{\mathcal{U}(x, r_x), \mathcal{S}(\mathit{CLS}_{SK})\}(C_x)$.

Further, for a commitment C'_x, $\mathcal{U}$ can prove the knowledge of (x, r'_x, e, s, v) [5, Figure 2], such that (x, r'_x) is an opening of C'_x, and (e, s, v) is a valid signature on x, where x and σ are hidden by the zero-knowledge property of the protocol. An auxiliary commitment scheme (g, h, n) is required, where n is the modulus used in the CLS_{PK}. We denote this protocol as: $\mathtt{PoKSigOnCommit}\{(\tilde{x}, \tilde{r}'_x, \tilde{\sigma}) : C'_x = a^x b^{r'_x} \wedge \mathit{CLS}.\mathtt{Verify}(\tilde{x}, \tilde{\sigma}, \mathit{CLS}_{PK})\}$.

This signature scheme can be extended to sign message tuples $(x_1, \ldots, x_k)$ by introducing k bases a_i [5]. Also, the pair of protocols above can be extended to support multiple messages, and selective message disclosure. For instance, we denote by $\mathtt{SigOnCommit}\{\mathcal{U}(\tilde{x}_1, \tilde{r}_{x_1}); \mathcal{S}(\mathit{CLS3}_{SK})\}(C_{x_1}, x_2, x_3)$ a protocol to generate a signature on a 3-tuple (x_1, x_2, x_3), where the message x_1, is (supposedly) blinded by a commitment C_{x_1}, and two messages x_2 and x_3 are disclosed in clear. Similarly, by $\mathtt{PoKSigOnCommit}\{(\tilde{x}_3, \tilde{r}_{x_3}, \tilde{\sigma}) : C_{x_3} = a_3^{\tilde{x}_3} b^{\tilde{r}_{x_3}} \wedge \mathit{CLS3}.\mathtt{Verify}(x_1, x_2, \tilde{x}_3, \tilde{\sigma}, \mathit{CLS3}_{PK})\}$ we denote the corresponding PoK that $\mathcal{U}$ knows a signature σ on a tuple (x_1, x_2, x_3), where x_1 and x_2 are disclosed to the verifier, but x_3 is kept blinded.

5.2 Construction

The components of our construction are two instances of the CL signature scheme: CLS, for messages in $[0, 2^{\ell_m})$, and $\mathit{CLS3}$, for messages in $[0, 2^{\ell_m})^3$.

Setup(1^κ). The vendor $\mathcal{V}$ generates an instance of the $\mathit{CLS3}$ signature scheme: $(\mathit{CLS3}_{PK}, \mathit{CLS3}_{SK}) := ((a_1, a_2, a_3, b, c, n), p) \leftarrow \mathit{CLS3}.\mathtt{Setup}(1^\kappa)$, and the CLS signature scheme: $(\mathit{CLS}_{PK}, \mathit{CLS}_{SK}) := ((\hat{a}, \hat{b}, \hat{c}, \hat{n}), \hat{p}) \leftarrow \mathit{CLS}.\mathtt{Setup}(1^\kappa)$. It is assumed that $\mathit{CLS3}$ and CLS have the same parameters $\ell_n, \ell_m, \ell_e, \ell$, and ℓ_s. Additionally, $\mathcal{V}$ generates two instances of the CS by computing $g, h \in_R \mathrm{QR}_n$, and $\hat{g}, \hat{h} \in_R \mathrm{QR}_{\hat{n}}$. These commitment schemes are only used in the $\mathtt{PoKSigOnCommit}$ protocol. Finally, $\mathcal{V}$ initializes a counter on sequence numbers: $\chi_{sq} \leftarrow 1$, stores $\mathit{SK} := (\mathit{CLS3}_{SK}, \mathit{CLS}_{SK})$, publishes $\mathit{PK} := (\mathit{CLS3}_{PK}, g, h, \mathit{CLS}_{PK}, \hat{g}, \hat{h})$, and creates an empty database DB of coupon identifiers.

Issue. In this protocol (Figure 2) the user $\mathcal{U}$ interacts with the vendor $\mathcal{V}$ to obtain k coupons with objects ob_i, for $0 \leq i < k$. First, $\mathcal{V}$ chooses a new sequence number $sq_0 \leftarrow \chi_{sq}$, updates the counter $\chi_{sq} \leftarrow \chi_{sq} + k + 1$, computes $\sigma'_0 \leftarrow \mathit{CLS}.\mathtt{Sign}(sq_0, \mathit{CLS}_{SK})$, and sends both sq_0 and σ'_0 to $\mathcal{U}$. Then, $\mathcal{U}$ randomly chooses k coupon identifiers id_i, for $0 \leq i < k$, and commits to them by computing the commitments C_{id_i}, which are sent to $\mathcal{V}$. Afterwards, for each i, $0 \leq i < k$, $\mathcal{U}$ executes the $\mathtt{SigOnCommit}$ protocol to obtain a $\mathit{CLS3}$ signature σ_i on $(id_i, ob_i, sq_0 + i)$, where id_i is kept blinded in C_{id_i}, and $ob_i, sq_0 + i$ are known by the $\mathcal{V}$. Notice that only the first coupon is redeemable.

Redeem. In the redeem protocol (Figure 3) $\mathcal{U}$ selects her next unused redeemable coupon $(id_i, ob_i, sq_i, \sigma_i, \sigma'_i)$ from her MC, commits to sq_i via $C_{sq_i} \leftarrow a_3^{sq_i} b^{r_{sq_i}}$, $C'_{sq_i} \leftarrow \hat{a}^{sq_i} \hat{b}^{r'_{sq_i}}$ using the appropriate moduli n and $\hat{n}$ of the two signature schemes, and sends id_i, ob_i, C_{sq_i}, and C'_{sq_i} to $\mathcal{V}$. The vendor checks that id_i

Common input: public keys $CLS_{PK} = (\hat{a}, \hat{b}, \hat{c}, \hat{n})$, $CLS3_{PK} = (a_1, a_2, a_3, b, c, n)$, number of single coupons k, object identifiers ob_i, $i = 0, \ldots, k-1$
User's input: –
Vendor's input: private keys $CLS_{SK} = \hat{p}$, $CLS3_{SK} = p$

```
User $\mathcal{U}$                                                  Vendor $\mathcal{V}$
                                                       $sq_0 \leftarrow \chi_{sq}$; $\chi_{sq} \leftarrow \chi_{sq} + k + 1$;
                                                       $\sigma'_0 \leftarrow CLS.\mathtt{Sign}(sq_0, CLS_{SK})$;
                          $\xleftarrow{\quad sq_0,\ \sigma'_0 \quad}$
$CLS.\mathtt{Verify}(sq_0, \sigma_0, CLS_{PK}) \overset{?}{=} true$
for each $i = 0, \ldots, k-1$ do
    $id_i \in_R (0, 2^{\ell_m})$;
    $r_{id_i} \in_R (0, 2^{\ell_n})$;
    $C_{id_i} \leftarrow a_1^{id_i} b^{r_{id_i}}$;
end do;
                          $\xrightarrow{\quad C_{id_0}, \ldots, C_{id_{k-1}} \quad}$
for each $i = 0, \ldots, k-1$ do
    $\sigma_i \leftarrow \mathtt{SigOnCommit}\{\mathcal{U}(id_i, r_{id_i}) : \mathcal{V}(CLS3_{SK})\}(C_{id_i}, ob_i, sq_i)$
                          $\longleftrightarrow$
    $sq_{i+1} \leftarrow sq_i + 1$;                               $sq_{i+1} \leftarrow sq_i + 1$;
end do;
```

Fig. 2. Issue Protocol

is not in the database, and inserts it. Then, $\mathcal{U}$ proves that C_{sq_i} and C'_{sq_i} are commitments to the same sequence number sq_i. Then, $\mathcal{U}$ uses `PoKSigOnCommit` to prove in zero knowledge that she knows a *CLS3* signature σ_i on the tuple (id_i, ob_i, sq_i) without disclosing σ_i. Additionally, $\mathcal{U}$ proves to $\mathcal{V}$ the knowledge of a *CLS* signature σ'_i on sq_i, without disclosing any useful information about it to $\mathcal{V}$. Finally, if every PoK succeeded, $\mathcal{U}$ obtains a signature σ'_{i+1} on $sq_{i+1} := sq_i + 1$, i.e., her next coupon becomes redeemable.

In the description above, it is assumed that $\mathcal{U}$ always outputs *rej* in case any obtained signature is invalid. Similarly, $\mathcal{V}$ must output *rej* in case any PoK fails.

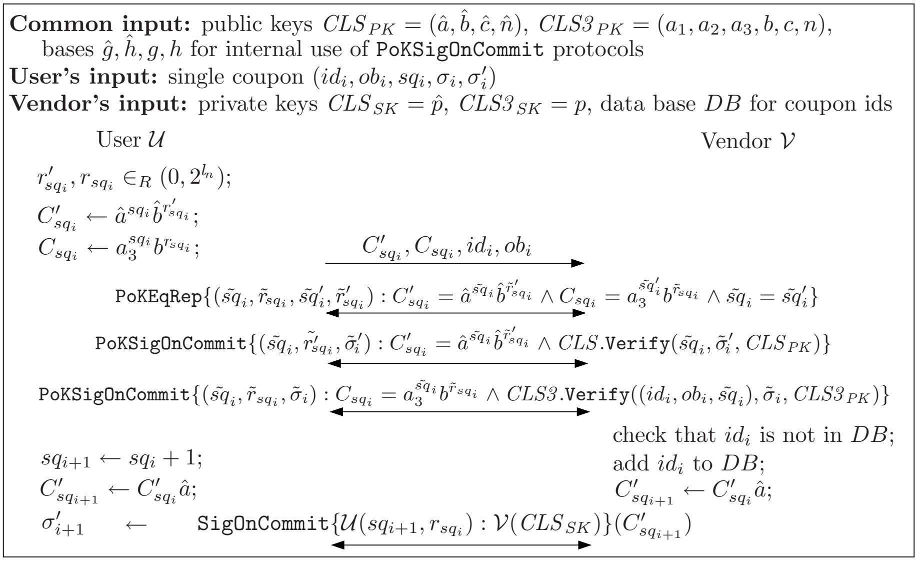
5.3 Security Proofs

In this section we present a number of theorems stating the properties of our scheme. Due to space restrictions we omit some proofs.

Theorem 2. *The MCS proposed in Section 5 is correct. (Proof omitted)*

Theorem 3. *The MCS proposed in Section 5 is unsplittable.*

Proof (Sketch). Assume $\mathcal{A}$ is an adversary against unsplittability. It is possible to construct an algorithm $\mathcal{B}$, which outputs a forgery to one of the signatures *CLS3* or *CLS* with at least half the success probability of $\mathcal{A}$ (minus some negligible term). $\mathcal{B}$ simulates the interface I_1, and must answer the queries made by $\mathcal{A}$. The only steps which $\mathcal{B}$ cannot trivially simulate are those which require the



Common input: public keys $CLS_{PK} = (\hat{a}, \hat{b}, \hat{c}, \hat{n})$, $CLS3_{PK} = (a_1, a_2, a_3, b, c, n)$, bases $\hat{g}, \hat{h}, g, h$ for internal use of `PoKSigOnCommit` protocols

User's input: single coupon $(id_i, ob_i, sq_i, \sigma_i, \sigma'_i)$

Vendor's input: private keys $CLS_{SK} = \hat{p}$, $CLS3_{SK} = p$, data base DB for coupon ids

User $\mathcal{U}$ Vendor $\mathcal{V}$

$r'_{sq_i}, r_{sq_i} \in_R (0, 2^{l_n})$;

$C'_{sq_i} \leftarrow \hat{a}^{sq_i} \hat{b}^{r'_{sq_i}}$;

$C_{sq_i} \leftarrow a_3^{sq_i} b^{r_{sq_i}}$; $\xrightarrow{C'_{sq_i}, C_{sq_i}, id_i, ob_i}$

$\mathtt{PoKEqRep}\{(\tilde{sq}_i, \tilde{r}_{sq_i}, \tilde{sq}'_i, \tilde{r}'_{sq_i}) : C'_{sq_i} = \hat{a}^{\tilde{sq}_i} \hat{b}^{\tilde{r}'_{sq_i}} \wedge C_{sq_i} = a_3^{\tilde{sq}'_i} b^{\tilde{r}_{sq_i}} \wedge \tilde{sq}_i = \tilde{sq}'_i\}$ $\longleftrightarrow$

$\mathtt{PoKSigOnCommit}\{(\tilde{sq}_i, \tilde{r}'_{sq_i}, \tilde{\sigma}'_i) : C'_{sq_i} = \hat{a}^{\tilde{sq}_i} \hat{b}^{\tilde{r}'_{sq_i}} \wedge CLS.\mathtt{Verify}(\tilde{sq}_i, \tilde{\sigma}'_i, CLS_{PK})\}$ $\longleftrightarrow$

$\mathtt{PoKSigOnCommit}\{(\tilde{sq}_i, \tilde{r}_{sq_i}, \tilde{\sigma}_i) : C_{sq_i} = a_3^{\tilde{sq}_i} b^{\tilde{r}_{sq_i}} \wedge CLS3.\mathtt{Verify}((id_i, ob_i, \tilde{sq}_i), \tilde{\sigma}_i, CLS3_{PK})\}$ $\longleftrightarrow$

check that id_i is not in DB;

$sq_{i+1} \leftarrow sq_i + 1$; add id_i to DB;

$C'_{sq_{i+1}} \leftarrow C'_{sq_i} \hat{a}$; $C'_{sq_{i+1}} \leftarrow C'_{sq_i} \hat{a}$;

$\sigma'_{i+1} \leftarrow \mathtt{SigOnCommit}\{\mathcal{U}(sq_{i+1}, r_{sq_i}) : \mathcal{V}(CLS_{SK})\}(C'_{sq_{i+1}})$ $\longleftrightarrow$

Fig. 3. Redeem Protocol

generation of a signature. To accomplish this he has black box access to $\mathcal{A}$, and access to two signature oracles $CLS3.\mathtt{Sign}(\cdot, CLS3_{SK})$ and $CLS.\mathtt{Sign}(\cdot, CLS_{SK})$ (for two randomly chosen secret keys $CLS3_{SK}$ and CLS_{SK} unknown to $\mathcal{B}$). Each time $\mathcal{B}$ must sign a message sq_0 in clear, he simply queries the CLS oracle.

The execution of the `SigOnCommit` protocol, in both the issue and redeem procedures, can be simulated towards $\mathcal{A}$ as described in the proof of [5, Lemma 6.1], where the actual signature computation is outsourced to the corresponding signature oracle.

Because of the soundness of every zero-knowledge PoK in the `Issue` and `Redeem` protocols, we can assume that during the protocol executions, $\mathcal{B}$ can extract (by using rewinding) all the witnesses for each PoK from $\mathcal{A}$. This allows $\mathcal{B}$ to obtain the attributes of all coupons, both issued and redeemed.

It is possible to prove that the number of unused redeemable coupons provided to $\mathcal{A}$ in the unsplittability game is at most χ_M (here it is important that $\mathcal{B}$ updates the counter χ_{sq} to avoid signing the same sequence number twice). In the last part of the unsplittability game $\mathcal{A}$ is able to redeem $\chi_M + 1$ coupons. Hence $\mathcal{B}$ is able to extract from $\mathcal{A}$ the information of $\chi_M + 1$ redeemable coupons $(id_i, ob_i, sq_i, \sigma_i, \sigma'_i)$, for $0 \leq i \leq \chi_M$. In particular, at least one of these coupons has a signature σ_i on (id_i, ob_i, sq_i) or σ'_i on sq_i, which was not queried to the respective signature oracle and therefore is an existential forgery of one of the signature schemes. In order to identify the forgery, $\mathcal{B}$ stores every signature-message pair queried to the signature oracles.

Theorem 4. *The MCS proposed in Section 5 is unforgeable.*

Proof. The theorem trivially follows from Theorems 1 and 3.

Coupon objects are useful features that unavoidably come at a price: they can be trivially used by the vendor to link protocol runs (e.g. by assigning unique coupon objects to each user). Hence, our construction does not meet unlinkability as in Definition 3. However, in practice, if there are many redeemable coupons at any time for each possible coupon object, then this information by itself does not substantially harm privacy.

Theorem 5. *The MCS of Section 5 restricted to constant coupon objects is unlinkable.*

Proof (Sketch). Wlog assume we can guess the users $\mathcal{U}_0$ and $\mathcal{U}_1$ challenged by $\mathcal{A}$. The proof is based on the existence of simulators for each one of the proofs of knowledge employed in the protocols, and can be organized as a sequence of games [16]. We can construct a series of modified games from the unlinkability game, by substituting, one by one, every PoK and every commitment used in the `Issue` and `Redeem` protocols executed by the users $\mathcal{U}_0$ and $\mathcal{U}_1$. The hiding property of the commitment scheme can be used to replace the commitments by random values. In each transition, the adversary's success probability is modified only by a negligible amount.

In the last game, the adversary's view regarding the users $\mathcal{U}_0$ and $\mathcal{U}_1$ is completely simulated, thus his success probability is exactly 1/2. This implies that his linkability advantage for the original game is negligible.

Complexity and Extensions. The computation and communication complexity of the issue protocol is linear in k, while the redemption complexity is constant in k. This improves the complexity of the scheme in [7], which is linear for both issue and redeem, but is less efficient than the scheme in [11], which has constant complexity for both protocols. However, for many applications the scheme in [11] is less practical than ours, because it lacks specific attributes per coupon, and it only offers weak unsplittability.

It is possible to extend the scheme by adding additional attributes to each coupon. For instance, we can easily implement validity periods by adding two attributes t^a and t^b, such that a coupon is only valid if a publicly known time variable belongs to the interval $[t^a, t^b]$. Furthermore, by using standard zero-knowledge interval protocols [2], it is possible for a user to prove that the coupon is valid at some precise date/time, without disclosing either t^a or t^b. Note that this has a similar effect on unlinkability as coupon objects.

6 Conclusion and Future Work

In this paper we introduced a privacy-protecting multi-coupon system, which improves previous proposals with regard to various aspects: better efficiency than [7], weaker assumptions than [11], and stronger security requirements. In particular, we provide an improved security model with a stronger definition of unsplittability, which discourages sharing of multi-coupons without relying on the all-or-nothing principle. Unlike alternative approaches, our scheme does not encode valuable information into the coupons to dissuade users from sharing them. Therefore, it can be considered more privacy-friendly. Moreover, it can be extended with additional attributes such as validity periods.

Some open problems are to design a *MCS* with the properties above, but in which coupons can be redeemed in arbitrary order, and to develop *MCS*s for more general settings (e.g. multiple collaborating vendors).

References

1. Blundo, C., Cimato, S., De Bonis, A.: Secure e-coupons. Electronic Commerce Research 5(1), 117–139 (2005)
2. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
3. Brands, S.: A technical overview of digital credentials. research report (February 2002), `http://www.xs4all.nl/#brands/`
4. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
5. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, Springer, Heidelberg (2003)
6. Canard, S., Gouget, A., Hufschmitt, E.: A handy multi-coupon system. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 66–81. Springer, Heidelberg (2006)
7. Chen, L., Enzmann, M., Sadeghi, A.-R., Schneider, M., Steiner, M.: A privacy-protecting coupon system. In: Patrick, A.S., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 93–108. Springer, Heidelberg (2005)
8. Damgård, I., Fujisaki, E.: A statistically hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, Springer, Heidelberg (2002)
9. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004)
10. Lipmaa, H.: On diophantine complexity and statistical zero–knowledge arguments. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003)
11. Nguyen, L.: Privacy-protecting coupon system revisited. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, Springer, Heidelberg (2006)
12. Nguyen, L., Safavi-Naini, R.: Dynamic k-times anonymous authentication. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 318–333. Springer, Heidelberg (2005)
13. Odlyzko, A.: Privacy, economics, and price discrimination on the internet. In: ICEC 2003. Proceedings of the 5th international conference on Electronic commerce, pp. 355–366. ACM Press, New York (2003)

14. Park, K., Gómez, M.: The coupon report: A study of coupon discount methods. Technical report, Department of Applied Economics and Management, Cornell University (2004), http://aem.cornell.edu/research/researchpdf/rb0407.pdf
15. Persiano, P., Visconti, I.: An efficient and usable multi-show non-transferable anonymous credential system. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, Springer, Heidelberg (2004)
16. Shoup, V.: Sequences of games: A tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), http://eprint.iacr.org/
17. Syverson, P.F., Stubblebine, S.G., Goldschlag, D.M.: Unlinkable serial transactions. In: FC 1997. LNCS, vol. 1318, pp. 39–56. Springer, Heidelberg (1997)