A Privacy-Protecting Coupon System

Liqun Chen[1], Matthias Enzmann[2,⋆], Ahmad-Reza Sadeghi[3],
Markus Schneider[2], and Michael Steiner[4]

[1] HP Labs, Filton Road, Stoke Gifford, Bristol BS34 8QZ, United Kingdom
liqun.chen@hp.com

[2] Fraunhofer Gesellschaft (FhG), Institute for Secure Information Technology (SIT),
Dolivostr. 15, D-64293 Darmstadt, Germany
{matthias.enzmann, markus.schneider}@sit.fraunhofer.de

[3] Ruhr-University Bochum,
Universitätsstr. 150, D-44780 Bochum, Germany
sadeghi@crypto.rub.de

[4] IBM T.J. Watson Research Center,
P.O. Box 704, Yorktown Heights, NY 10598, USA
msteiner@watson.ibm.com

Abstract. A coupon represents the right to claim some service which is typically offered by vendors. In practice, issuing bundled multi-coupons is more efficient than issuing single coupons separately. The diversity of interests of the parties involved in a coupon system demands additional security properties beyond the common requirements (e.g., unforgeability). Customers wish to preserve their privacy when using the multi-coupon bundle and to prevent vendors from profiling. Vendors are interested in establishing a long-term customer relationship and not to subsidise one-time customers, since coupons are cheaper than the regular price. We propose a secure multi-coupon system that allows users to redeem a predefined number of single coupons from the same multi-coupon. The system provides unlinkability and also hides the number of remaining coupons of a multi-coupon from the vendor. A method used in the coupon system might be of independent interest. It proves knowledge of a signature on a message tuple of which a single message can be revealed while the remaining elements of the tuple, the index of the revealed message, as well as the signature remain hidden.

1 Introduction

Today, coupons appear to be useful means for vendors to attract the attention of potential customers. Usually, coupons give the customer a financial incentive to purchase at a specific vendor. The purpose of coupons is many-fold. For instance, they can be used to draw the attention of customers to a newly opened shop or to prevent customers from buying at a competitor's shop [29]. Of course, coupons can also be purchased by

[⋆] The author was supported in part by the German Federal Ministry of Education and Research under grant 01AK706C, project *Premium*.

A.S. Patrick and M. Yung (Eds.): FC 2005, LNCS 3570, pp. 93–108, 2005.

customers, e.g., gift certificates. Even drug prescriptions from a doctor can be seen as a kind of a coupon.

In general, a coupon is a representation of the right to claim some good or service, usually from the party that issued the coupon. The types of coupons mentioned before can, in general, be redeemed only once, i.e., the coupon is invalidated after the service or good has been claimed. However, there are also coupons which can be redeemed more than once, such as a coupon book of a movie theater, where customers pay, e.g., for 9 movies and are entitled to see 10. We call such coupons *multi-coupons*. In this paper, we are particularly interested in this type of coupons.

Typically, a real-world multi-coupon of value m is devalued by crossing out some field or by detaching a part of it. Offering such coupons can be beneficial for the issuing party, e.g., a movie theater. First, customers pay in advance for services or goods they have not claimed yet. Second, they are locked-in by the issuer/vendor, i.e., they are unlikely to switch to another vendor to purchase the same or similar service or good as long as they have not redeemed all their coupons. Hence, multi-coupons can also be seen as a kind of loyalty program since they are specific to some vendor and induce loyalty, at least, as long as the customer has coupons left to spend.

Clearly, vendors are interested in creating loyalty and hence, it is likely that we are going to see such coupon systems in the Internet, too. In fact, introducing such a coupon system might be even more valuable to Internet vendors than to their real-world counterparts. Since, from the customers' viewpoint, *a priori* all vendors, offering a certain good or service, look alike and can be reached as easily as their competitors.

This is in contrast to the real world where time and effort is required to go to physical stores to acquire information on that store's products [23]. In the real world, additional barriers may exist, such as physical distance or some kind of relationship to shop personnel, that incur indirect switching costs for customers [21]. In summary, it can be expected that in absence of notable switching costs customer fluctuation is higher in the Internet than in the real world because there is little that keeps customers from buying at a competitor's site whenever this competitor offers a better price. Thus, it is in the vendor's interest to introduce switching costs in order to retain an installed base of customers [30].

1.1 Desirable Properties for Coupon Systems

At first, introducing a coupon system looks like a win-win situation, since both parties seem to benefit from such a coupon system. Vendors have a means to create a loyal customer base and customers value the financial benefit provided by coupons. However, since a customer normally redeems her coupons in different transactions, a multi-coupon can be used as a means to link transactions, and thus, to allow a vendor to create a record of the customer's past purchases. Such customer information might be exploited for data mining, to infer new customer data, customer profiling, promotion of new products, price discrimination, etc. [24]. Thus, if through usage of the coupon system customers expect a misuse of their personal data, e.g., by using it to create profiles for price discrimination [16], they are more likely to decline the coupon system. According to [19, 20] privacy is a concern to Internet users, especially when it comes to electronic commerce scenarios. Hence, a prudent vendor should take these concerns into account when planning to offer a coupon system.

In order to rule out privacy concerns of customers from the start, vendors might want to introduce a coupon system that does not infringe their customers' privacy. Thus, a coupon should disclose as little information as possible. For instance, a multi-coupon should only give vendors an indication that it is still valid, i.e., that at least one coupon is not spent, instead of disclosing the number of unspent coupons. Such a property could be useful in sensitive areas, e.g., in health care scenarios, where a multi-coupon can be used as a prescription for a certain number of doses of some medicine. In this case, the pharmacist would deduct a single coupon from the multi-coupon and may only detect if the prescription has been used up. Also in welfare, paper-based checks or food stamps could be replaced by electronic coupons. In fact, recently, the U.S. announced to replace their paper-based food stamp program with electronic benefits and debit cards [26]. However, this electronic program does not protect the privacy of recipients, since the cards are processed similar to ordinary debit cards.

For vendors, in addition to common security requirements such as unforgeability, there are other requirements which are specific to a coupon system. As mentioned before, a vendor's driving reason for offering a coupon system is to establish a long term relationship with customers. However, customers may be interested in sharing a multi-coupon, i.e., each customer obtains and redeems a fraction of the coupons in the multi-coupon. Moreover, this behaviour allows them, e.g., to sell coupons on an individual basis for a cheaper price[1], e.g., to one-time customers who otherwise would have purchased full-price services or goods. Thus, ideally, vendors wish to prevent customers from *splitting* their coupons.

To illustrate splitting, we consider the following variants as examples of real-world multi-coupons. The first variant, being a coupon book with detachable coupons and the second one being a multi-coupon where spent coupons are crossed out, i.e., coupons cannot be detached. The coupon book can be easily shared by a group of customers, since each customer can detach its share of coupons from the coupon book and each coupon may be independently redeemed by a different customer. In the second variant, the multi-coupon must be given to the vendor *as a whole* to allow him to devalue the multi-coupon by crossing out one of the coupons. Hence, in this variant, individual coupons cannot be split and redeemed separately and independently as in the first variant.

Nevertheless, even in the multi-coupon scenario with non-detachable coupons some kind of sharing is possible if we transfer it to the digital world. Since digital coupons can be easily copied, colluding customers may jointly purchase a multi-coupon, distribute copies of it among each other, and agree to redeem only the share of the coupons for which each of them paid for. In this scenario, however, customers have to fully trust each other that none of them redeems more than its share of coupons. Since each of the colluders owns a copy of the multi-coupon this means that every colluder has full control of all single coupons. Hence, each of them could redeem single coupons of other colluders without their knowledge. A colluder deceived in such a way would only learn

[1] Recall that a multi-coupon for m goods is sold for the price of $m - k$ goods, $k \geq 1$.

about it when he or she tries to redeem a single coupon and the vendor rejects it because it was already spent. Thus, it seems less likely that multi-coupons are traded between customers.

In this context, another scenario with multi-coupons is possible where trust is only one-way. If customer A buys a multi-coupon, uses up, say, half of the coupons and sells the remaining half of the coupons to customer B then A does not have to trust B. Only B has to trust A that he indeed received the purported half of the coupons. There is nothing that really can stop A from doing so, neither in a real-world scenario with paper coupons nor in the digital scenario, unless (a) the multi-coupon contains information that ties it to A's identity and which the vendor may verify (b) customer A has a strong incentive to keep the multi-coupon, e.g., because some private and/or "valuable" information is encoded into the multi-coupon.

We do not pursue any of these two approaches since, first, we do not want to identify customers because this may violate their privacy and, second, encoding valuable information seems to be unsatisfactory as well because encoding a "valuable" secret implies that such a secret exists and that a customer is willing to encode it into a, potentially, considerably less valuable multi-coupon. Instead, we employ *all-or-nothing sharing* which has been used in other works before [3, 8] to *discourage* users from sharing / disclosing certain data, such as private credential information.

In case of a multi-coupon, all-or-nothing means that a customer cannot give away or sell any single coupon from its multi-coupon without giving away all other single coupons — this includes used and unused single coupons alike. Therefore, our scheme is comparable with the real-world multi-coupon example from above where used coupons are crossed out. The difference is that in the digital world one may effortlessly create identical copies of a multi-coupon while in the real world creating exact hardcopies of such a coupon may require some effort.

1.2 Overview of Our Coupon System

The coupon system proposed here can be viewed as a digital counterpart to the real-world multi-coupon with non-detachable coupons, as mentioned before. In our coupon system, a multi-coupon M is a signature on a tuple X where $X = (x_1, \ldots, x_m)$. In the system specification, we denote a set of coupons by M and a single coupon by $x \in \{x_1, \ldots, x_m\}$.

In the coupon issue phase, a user first convinces a vendor that she knows X without revealing the values of X. Then, the verifier issues the coupon M by "blindly" signing X, i.e., $M := Sign(X)$, and sending M to the user. Here we make use of the Camenisch and Lysyanskaya (CL) signature scheme [9].

When redeeming a single coupon x, the user reveals x to the vendor and proves that she is in possession of a valid multi-coupon $M = Sign(X)$ and $x \in \{x_1, \ldots, x_m\}$. The vendor then checks if x is in a list of used coupons. If it is not, the vendor accepts x and puts it in the list. Beside satisfying common security requirements, the scheme has the following properties: The vendor is not able to trace x back to M or to link two redemptions since M is never given back to the vendor and the single coupons x are independent of each other. Furthermore, the vendor does not learn anything about the status of the multi-coupon, i.e., how many single coupons are left in the multi-coupon.

Concerning the vendor's requirement, the scheme does not allow users to split a multi-coupon without sharing all values $(x_1, \ldots, x_m)$.

A method used in the coupon system might be of independent interest. It proves knowledge of the CL signature M on a message tuple $X := (x_1, \ldots, x_m)$, of which an arbitrary single message x_j can be revealed while the remaining elements of the tuple, the revealed message's index, j, and the signature M remain hidden.

2 Related Work

At first it may seem that the coupon system can be easily realised using an existing payment system or credential system which supports m-showable credentials or at least one-showable credentials of which m can be obtained. However, none of these systems satisfied all the requirements of the coupon system we had in mind, or could only satisfy them in an inefficient manner. In addition, some of the systems discussed below require the existence of a trusted third party to issue certificates of some sort. We do not have such a requirement.

The payment system of Chaum [11] as well as the one of Brands [2] use digital coins which can be anonymously spent. Withdrawal and spending of coins is roughly the same as issuance and redemption of single coupons. However, using m single-spendable digital coins as a multi-coupon easily allows splitting of the multi-coupon. Even if we would use multi-valued coins such that one unit of an m-coin can be spent and an $m-1$ coin is returned, we would still not have the coupon system that we have in mind, since the number of remaining coins is disclosed to the vendor. In the coin system of Ferguson [17] a multi-coin is introduced that can be spent m times. However, when paying with the same multi-coin the vendor learns the remaining value of the coin and, in addition, transactions are linkable.

Okamoto and Ohta [25] proposed a scheme which resembles our coupon system in the sense that they use a multiple blind signature to issue a "large" coin which is comprised of "smaller" coins, or rather, can be subdivided into smaller ones. However, subdividability in their system is considered a feature while in a coupon system this translates to splitting and, hence, is less desirable. In [12] and [31], Chen and Verheul, respectively, proposed credential systems where the credentials are multi-showable, i.e., can be shown for an unlimited number of times. The credentials obtained through both systems are intended for pseudonymous usage, thus, our requirements for unlinkable redemptions and m-redeemability are not satisfied.

In the work of Brands [3], attribute certificates were proposed that allow selective showing of individual attributes. These attributes are normally multi-showable but can be made m-showable, however, then different transactions become linkable. Persiano and Visconti [27] used some of the ideas of [3] to build a credential system which is multi-showable and does not allow to link different showings of a credential. Still, showings of credentials cannot be limited.

An anonymous credential system where credentials can be made one-showable was proposed by Camenisch and Lysyanskaya [8]. Through this system, a user may obtain m one-show credentials which can be regarded as single coupons. However, this approach is not very efficient when used in a coupon system, since a credential generation protocol must be run for each credential and the credentials can be shared by different users and independently shown[2]. This means, when applied as a coupon system, coupons can be independently spent and splitting is easily possible.

The aspect of technically supporting loyalty in commercial applications has also been explored before. Maher [22] proposed a framework to introduce loyalty points, however, the privacy aspect was not an issue there. Enzmann et al. [15] proposed a counter for a privacy-friendly, point-based loyalty system, where users anonymously obtained points for their purchases. Finally, Wibowo et al. [32] proposed a loyalty system, however, based on pseudonyms, thus, providing a weaker form of privacy.

3 Model

The coupon system considered here involves mainly two parties, a customer $\mathcal{U}$ (user) and a vendor $\mathcal{V}$. The system itself is comprised of an *issue* protocol and a *redeem* protocol which both are carried out between $\mathcal{U}$ and $\mathcal{V}$. The output of the issue protocol is a multi-coupon M for $\mathcal{U}$ and the result of the redeem protocol is a spent single coupon for $\mathcal{V}$ and a multi-coupon devalued by one single coupon for $\mathcal{U}$. Next, we state the main security requirements for the involved parties.

3.1 Requirements

In the following, we will use the notation $M \rightsquigarrow N$ to indicate that multi-coupon N is a successor of multi-coupon M. That is, N has strictly less coupons left to spent than M and both originate from the same initial multi-coupon. Given two multi-coupons M and N, if either $M \rightsquigarrow N$ or $N \rightsquigarrow M$ we say that M and N are *related*.

Unforgeability. It must be infeasible to create new multi-coupons, to increase the number of unspent coupons, or to reset the number of spent coupons.

Double-spending detection. A vendor must be able to detect attempts of redeeming 'old' coupons that have already been redeemed. This means, given two runs of the redeem protocol, where a single coupon x is deducted from multi-coupon M and y is deducted from N, the vendor must be able to decide if $x = y$.

Redemption limitation. An m-redeemable coupon M may not be accepted by the vendor more than m times.

Protection against splitting. A coalition of customers $\mathcal{U}_i$ should not be able to split an m-redeemable multi-coupon M into (disjoint) s_i-redeemable shares M_i with $\sum_i s_i \leq m$ such that M_i can only be redeemed by customer $\mathcal{U}_i$ and none of the other customers $\mathcal{U}_j, j \neq i$, or a subset of them is able to redeem the share M_i or a part of it. We call this property *strong protection against splitting*.

A weaker form of this property is *all-or-nothing-sharing*[3]. This means that splitting is possible, however, only if customers trust each other not to spent (part of) the other's

[2] In [8] a solution was proposed to deal with this kind of lending. However, this solution hurts performance because it adds complexity and additional protocol runs to the basic scheme.

[3] This is similar to all-or-nothing-disclosure used in [3, 8] to prevent lending of credentials.

share M_i. Another way of putting this is to say that sharing M means sharing all m single coupons. We call this *weak protection against splitting*.

Unlinkability. It must be infeasible for vendors to link protocol runs of honest users. For this, we have to consider linking a run of an issue protocol to runs of corresponding redeem protocols and linking of any two redeem protocol runs.

(1) issue vs. redeem: Given a run of the issue protocol with output a multi-coupon M and given a redeem protocol run with output a devalued multi-coupon N, the vendor must not be able to decide if $M \rightsquigarrow N$.

(2) redeem vs. redeem: Given two runs of the redeem protocol with output two multi-coupons M, N, the vendor must not be able to decide if $M \rightsquigarrow N$ or $N \rightsquigarrow M$, i.e., he cannot tell if M and N are related or unrelated.

Minimum Disclosure. As a result of a redeem protocol run, the vendor may only learn of the single coupon being redeemed but not the number of remaining coupons. This already follows from the unlinkability requirement but we make it explicit here, nevertheless.

4 Building Blocks

In order to illustrate our coupon system, we first introduce the employed technical building blocks.

Throughout the paper, we will use the following notational conventions. Let $n = pq$ where $p = 2p' + 1$, $q = 2q' + 1$ and p', q', p, q are all primes. Denote the binary length of an integer I by ℓ_I. We require $\ell_p = \ell_n/2$. We denote the set of residues modulo n that are relatively prime to n by $\mathbb{Z}_n^*$ and the set of quadratic residues modulo n by QR_n, i.e., for all $a \in QR_n$ there exists $b \in \mathbb{Z}_n^*$ such that $a \equiv b^2 \bmod n$. By $a \in_R S$ we mean that a is chosen uniformly and at random from the set of integers S. For saving space, we omit the operation $\bmod n$ in the following specifications.

4.1 Commitment Scheme

A commitment scheme is a two-party protocol between a committer $\mathcal{C}$ and a receiver $\mathcal{R}$. In general, the scheme includes a *Commit* process and an *Open* process. In the first process, $\mathcal{C}$ computes a commitment C_x with a message x, such that x cannot be changed without changing C_x [4]. $\mathcal{C}$ then gives C_x to $\mathcal{R}$ and keeps x secret. In the second process, $\mathcal{C}$ opens C_x by revealing x.

The commitment scheme we employ is due to Damgård and Fujisaki (DF) [14] which is a generalization of the Fujisaki-Okamoto scheme [18]. We skip the basic DF scheme for committing to a single value x and proceed to the scheme where the commitment is to a message tuple $(x_1, x_2, \ldots, x_m)$.

Let $\langle h \rangle$ denote the group generated by $h \in_R QR_n$, and let $g_1, g_2, \ldots, g_m \in \langle h \rangle$. On secret input $X := (x_1, x_2, \ldots, x_m)$, where $x_i \in [0, 2^{\ell_x})$, and public input $PK := (g_1, \ldots, g_m, h, n)$, the commitment is $C_X := \prod_{i=1}^{m} g_i^{x_i} h^{r_X}$, where $r_X \in_R \mathbb{Z}_n$ are chosen at random.

4.2 Signature Scheme

The signature scheme stated in the following is a variant of the Camenisch and Lysyanskaya (CL) signature scheme [9] for signing a block of messages which was used before in [5]. The signed message is denoted by a tuple $X := (x_1, x_2, \ldots, x_m)$ where $x_i \in [0, 2^{\ell_x})$, $i = 1, \ldots, m$ and ℓ_x is a parameter for the message length.

Key Generation. Set the modulus n as described before. Choose $a_1, a_2, \ldots, a_m$, b, c $\in_R QR_n$ and output a public key $PK := (A, b, c, n)$ where $A := (a_1, a_2, \ldots, a_m)$ and a secret key $SK := (p, q, n)$.

Signing. On input $X := (x_1, x_2, \ldots, x_m)$, choose a random prime number $e \in_R [2^{\ell_e - 1}, 2^{\ell_e - 1} + 2^{\ell'_e - 1}]$ and a random number s of length ℓ_s, where ℓ'_e is the length of the interval that the e values are chosen from, ℓ_e is the length of the e values, ℓ_s is the length of the s value. Both the values ℓ_e and ℓ_s are dependent on a security parameter ℓ_ϕ, for the details see [5]. The resulting signature is the tuple (v, e, s) such that $c \equiv v^e a_1^{x_1} \cdots a_m^{x_m} b^s$. We denote this algorithm by: $(v, e, s) \leftarrow \mathit{Sign}_{(A,b,c,n,p)}(X)$.

Verification. In order to verify that (v, e, s) is a signature on $X := (x_1, x_2, \ldots, x_m)$, check that $c \equiv v^e a_1^{x_1} \cdots a_m^{x_m} b^s$ and also that $x_i \in [0, 2^{\ell_x})$, $i = 1, \ldots, m$, and $2^{\ell_e - 1} \geq e \geq 2^{\ell_e - 1} + 2^{\ell'_e - 1}$. We denote this algorithm by: $\mathit{ind} \leftarrow \mathit{Verify}_{(A,b,c,n)}(X, v, e, s)$, where $\mathit{ind} \in \{\mathit{accept}, \mathit{reject}\}$.

Remark 1. The CL signature scheme is separable, i.e., the signature (v, e, s) on X is also the signature on a sub-tuple of X if we change the public key accordingly. In the following, we use the notation $X \backslash (x_j)$ to denote the sub-tuple of X which is comprised of all of X's components but its j-th one, i.e., $X \backslash (x_j) = (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_m)$. Now, the signature on X under the public key (A, b, c, n) is the same as the signature on $X \backslash (x_j)$ under the public key $(A \backslash (a_j), b, c/a_j^{x_j}, n)$, i.e., $\mathit{Sign}_{(A,b,c,n,p)}(X) = (v, e, s) = \mathit{Sign}_{(A \backslash (a_j), b, c/a_j^{x_j}, n, p)}(X \setminus (x_j))$. This holds for any sub-tuple Y of X. We will use this property in our coupon system to redeem a single coupon from a multiple set of coupons.

Remark 2. As discovered in [7], the CL signature scheme has the property of randomisation, i.e., the signature (v, e, s) can be randomised to $(T = vb^w, e, s^* = s - ew)$ with an arbitrary w. From a verifier's point of view, (T, e, s^*) and (v, e, s) are equivalent since they both are signatures on X. This property benefits our scheme because a proof of knowledge of (T, e, s^*) can be done more efficiently than proving knowledge of (v, e, s) in an anonymous manner.

4.3 Proofs of Relations Between Committed Numbers

For the construction of our coupon system, we need several sub-protocols in order to prove certain relations between committed numbers [18, 10, 1, 13, 28]. These are proofs of knowledge (PoK) where the commitments are formed using the DF scheme. We will now briefly state the various protocols that we employ. The output of each of the protocols for the verifier is $\mathit{ind}_{\mathcal{V}} \in \{\mathit{accept}, \mathit{reject}\}$.

PoKRep. A prover $\mathcal{P}$ proves knowledge of a discrete logarithm representation (DL-Rep) modulo a composite to a verifier $\mathcal{V}$. Common inputs are a description of group $\mathcal{G}$, $PK := (g_1, \ldots, g_m, h)$ with $h, g_i \in \mathcal{G}$, and a commitment C. By this protocol, $\mathcal{P}$ convinces $\mathcal{V}$ of knowledge of $X := (x_1, \ldots, x_m)$, such that $C = \prod_{i=1}^{m} g_i^{x_i} h^r$.

PoKEqRep. A prover $\mathcal{P}$ proves to a verifier $\mathcal{V}$ knowledge of equality of representations of elements from possibly different groups $\mathcal{G}_1, \mathcal{G}_2$. Common inputs are $PK_1 := (g_1, \ldots, g_m, h)$, $g_i, h \in \mathcal{G}_1$, $PK_2 := (g'_1, \ldots, g'_m, h')$, $g'_i, h' \in \mathcal{G}_2$, commitments $C_1 \in \mathcal{G}_1$ and $C_2 \in \mathcal{G}_2$. By running the protocol, $\mathcal{P}$ convinces $\mathcal{V}$ of knowledge of $X := (x_1, \ldots, x_m)$ such that $C_1 = \prod_{i=1}^{m} g_i^{x_i} h^{r_1}$ and $C_2 = \prod_{i=1}^{m} g_i'^{x_i} h'^{r_2}$, i.e., $log_{g_i}(g_i^{x_i}) = log_{g'_i}(g_i'^{x_i})$ $(i = 1, \ldots, m)$.

PoKInt. A prover $\mathcal{P}$ proves to a verifier $\mathcal{V}$ knowledge of x and r such that $C = g^x h^r$ and $a \leq x \leq b$. Common inputs are parameters (g, h, n), the commitment C, and the integers a, b. We use a straightforward extension to the basic protocol, such that the proved knowledge is two tuples, instead of two values, and the interval membership of each component from a tuple, instead of one value. Within this extension, we denote $G := (g_1, g_2, ..., g_m)$, $H := (h_1, h_2, ..., h_l)$, $X := (x_1, x_2, ..., x_m)$, $R := (r_1, r_2, ..., r_l)$, and $C := \prod_{i=1}^{m} g_i^{x_i} \prod_{j=1}^{l} h_j^{r_j}$. By running the protocol, $\mathcal{P}$ proves to $\mathcal{V}$ knowledge of X and R, and the interval membership, $a \leq x_i \leq b$.

PoKOr. A prover $\mathcal{P}$ proves to a verifier $\mathcal{V}$ an OR statement of a commitment C, such that $C := (C_1, \ldots, C_m)$, where $C_i := \prod_{j \in \alpha_i} g_j^{x_{ij}} h^{r_i}$ and $\alpha_i \subseteq \{1, ..., m\}$, and $\mathcal{P}$ knows at least one tuple $\{x_{ij} \mid j \in \alpha_i\}$ for some undisclosed i. We denote the OR statement as $\bigvee_{i=1}^{m} C_i = \prod_{j \in \alpha_i} g_j^{x_{ij}} h^{r_i}$. Common inputs are C and parameters (G, n) where $G := (g_1, \ldots, g_m)$. By running the protocol, $\mathcal{P}$ proves to $\mathcal{V}$ knowledge of $\{x_{ij} \mid j \in \alpha_i\}$ without revealing the values x_{ij} and i. A number of mechanisms for proving the ”OR” statement have been given in [6, 13, 28].

PoK. Sometimes, we need to carry out two or more of the above protocols simultaneously, e.g., when responses to challenges have to be used in more than one validity check of the verifier to prove intermingled relations among commitments. Instead of giving concrete constructions of these protocols each time, we just describe their aim, i.e., what the verifier wants to prove. For this we apply the notation used, e.g., in [10]. For instance, the following expression

$$ind_{\mathcal{V}} \leftarrow PoK\{(\alpha, \beta) : C = g^{\alpha} h^{\beta} \;\wedge\; D = \hat{g}^{\alpha} \hat{h}^{\beta} \;\wedge\; 0 \leq \alpha < 2^k\}$$

means that knowledge of α and β is proven such that $C = g^{\alpha} h^{\rho}$ and $D = \hat{g}^{\alpha} \hat{h}^{\beta}$ holds, and α lies in the integer interval $[0, 2^k)$.

4.4 Blind Signatures and Signature Proof

BlindSign. Next we state a secure protocol for signing a blinded tuple, shown in Figure 1, which is based on [9, 5]. In this protocol, a user $\mathcal{U}$ obtains a signature from the signer $\mathcal{S}$ on a tuple $X := (x_1, x_2, \ldots, x_m)$ without revealing X to $\mathcal{S}$. We assume that $\mathcal{S}$ has the public key $PK := (A, b, c, n)$, the secret key $SK := p$, and public length

User $\mathcal{U}$		**Signer $\mathcal{S}$**
Common Input:	Verification key $PK := (A, b, c, n), A := (a_1, \ldots, a_m)$ Length parameters $\ell_x, \ell_e, \ell'_e, \ell_s, \ell_n, \ell_\phi$	
User's Input:	Message $X := (x_1, \ldots, x_m)$	
Signer's Input:	Factorisation of n : (p, q, n)	
choose $s' \in_R \{0,1\}^{\ell_n+\ell_\phi}$		
compute $D := \prod_{i=1}^m a_i^{x_i} b^{s'}$	$\xrightarrow{D}$	
	Run PoK $\{$ $(\xi_1, \ldots, \xi_m, \sigma) : D = \pm a_1^{\xi_1} \cdots a_m^{\xi_m} b^\sigma$ $\wedge$ for $i = 1, \ldots, m : \xi_i \in \{0,1\}^{\ell_x+\ell_\phi+2}$ $\wedge$ $\sigma \in \{0,1\}^{\ell_n+\ell_\phi+2}\} \rightarrow ind_{\mathcal{S}}$	
		check $ind_{\mathcal{S}} \overset{?}{=} accept$ choose $\hat{s} \in_R \{0,1\}^{\ell_s-1}$ compute $s'' := \hat{s} + 2^{\ell_s-1}$ choose $e \in_R (2^{\ell_e-1}, 2^{\ell_e-1} + 2^{\ell'_e-1})$
compute $s := s' + s''$;	$\xleftarrow{(v,e,s'')}$	compute $v := (c/(Db^{s''}))^{1/e}$
check $Verify_{(A,b,n)}(X, v, e, s) \overset{?}{=} accept$ [i.e., $c = v^e b^s \prod_{i=1}^m a_i^{x_i}$] output (v, e, s)		

Fig. 1. Protocol for blindly signing a tuple: $BlindSign$

parameters $\ell_n, \ell_x, \ell_e, \ell'_e, \ell_s$, and ℓ_ϕ which are parameters controlling the statistical zero-knowledge property of the employed PoK. $\mathcal{U}$'s input to the protocol is the message $X := (x_1, \ldots, x_m)$ for which $\mathcal{U}$ wants to obtain a signature.

Among the first steps, $\mathcal{U}$ computes the value $D := \prod_{i=1}^m a_i^{x_i} b^{s'}$ and sends it to $\mathcal{S}$. The next steps assure to $\mathcal{S}$ that $\mathcal{U}$ indeed knows the discrete logarithms of D with respect to the basis $(a_1, \ldots, a_m, b)$ respectively, and the interval of the committed values in D are selected correctly.

If all proofs are accepted, $\mathcal{S}$ chooses a prime e and computes $v := (c/(Db^{s''}))^{1/e} = (c/(\prod_{i=1}^m a_i^{x_i} b^{(s'+s'')}))^{1/e}$. At the end, $\mathcal{V}$ sends the resulting tuple (v, e, s'') to $\mathcal{U}$. Finally, $\mathcal{U}$ sets $s := (s' + s'')$ and obtains (v, e, s) as the desired signature on X. We denote this protocol for blindly signing a tuple by $(v, e, s) \leftarrow BlindSign_{(PK)}(X)$.

PoKSign. The next protocol, shown in Figure 2, is a zero-knowledge proof of a signature created in the $BlindSign$ protocol. The idea of this protocol is to convince a verifier $\mathcal{V}$ that a prover $\mathcal{P}$ holds a valid signature (v, e, s) on X satisfying $c \equiv v^e a_1^{x_1} \cdots a_m^{x_m} b^s$ without $\mathcal{V}$ learning anything of the signature but its validity. The common inputs are the same as in the $BlindSign$ protocol. $\mathcal{P}$'s secret input is the message X and the corresponding signature (v, e, s).

The protocol works as follows: $\mathcal{P}$ first randomises the signature components, v and s, by choosing w at random and computing $T := vb^w$ and $s^* = s - ew$. $\mathcal{P}$ sends only T to $\mathcal{V}$. Then, $\mathcal{P}$ proves to $\mathcal{V}$ his knowledge specified in PoK.

As discussed in Remark 2 of Section 4.2, in $\mathcal{V}$'s view, (T, e, s^*) is a valid signature on X, as is (v, e, s). The difference between them is that we are allowed to reveal the

Prover $\mathcal{P}$ **Verifier** $\mathcal{V}$

Common Input: Verification key $PK := (A, b, c, n), A = (a_1, a_2, \ldots, a_m)$
Length parameters $\ell_x, \ell_e, \ell'_e, \ell_\phi$

Prover's Input: Message $X := (x_1, \ldots, x_m),$ Signature (v, e, s)

choose $w \in_R \{0,1\}^{\ell_n + \ell_\phi}$

compute $T := vb^w$; $\xrightarrow{T}$

Run PoK $\{$ $(\xi_1, \ldots, \xi_m, \sigma, \epsilon) : c = \pm T^\epsilon a_1^{\xi_1} \cdots a_m^{\xi_m} b^\sigma$ $\wedge$
for $i = 1, \ldots, m : \xi_i \in \{0,1\}^{\ell_x + \ell_\phi + 2}$ $\wedge$
$(\epsilon - 2^{\ell_e}) \in \{0,1\}^{\ell'_e + \ell_\phi + 1}\}$ $\rightarrow ind_\mathcal{V}$
check $ind_\mathcal{V} \overset{?}{=} accept$

Fig. 2. Protocol for proving knowledge of a signature: $PoKSign$

value T to $\mathcal{V}$, but not the value v, because T is different in every proof. Therefore, to prove the signature with $c \equiv v^e \prod_{i=1}^m a_i^{x_i} b^s$ becomes one with $c \equiv T^e \prod_{i=1}^m a_i^{x_i} b^{s^*}$. Clearly, to prove the second equation is much simpler then the first one. PoK here performs the following three simple proofs in one go: (1) $PoKRep$: to prove knowledge of discrete logarithms of c ($\equiv T^e \prod_{i=1}^m a_i^{x_i} b^{s^*}$) with respect to the basis $(T, a_1, \ldots, a_m, b)$ respectively; (2) $PoKInt$: to prove the values $x_1, \ldots, x_m$ are within a right bound, i.e., for $i = 1, \ldots, m : x_i \in \{0,1\}^{\ell_x + \ell_\phi + 2}$; (3) $PoKInt$: to prove the value e is also within a right bound, i.e., $(e - 2^{\ell_e}) \in \{0,1\}^{\ell'_e + \ell_\phi + 1}$.

5 Construction of the Coupon System

In this section we propose a concrete scheme for a coupon system that allows issuance and redemption of multi-coupons. The scheme is comprised of two protocols, *Issue* and *Redeem*, which are carried out between a user $\mathcal{U}$ and a vendor $\mathcal{V}$, and an Initialisation algorithm.

Initialisation. $\mathcal{V}$ initialises the system by generating a key pair $PK = (A, b, c, n)$ where $A = (a_1, a_2, \ldots, a_m)$ and $SK = (p, q, n)$. The vendor keeps SK secret and publishes PK with length parameters ℓ_x, ℓ_e, ℓ'_e, ℓ_n, ℓ_s, and the security parameter ℓ_ϕ.

Issue. In the issue protocol, $\mathcal{U}$ chooses serial numbers $x_i \in_R \{1, \ldots, 2^{\ell_x} - 1\}$ $(i = 1, \ldots, m)$ and sets $X := (x_1, \ldots, x_m)$. Then $\mathcal{U}$ runs $(v, e, s) \leftarrow BlindSign_{(PK)}(X)$ with $\mathcal{V}$ to obtain a blind CL signature (v, e, s) on X. The tuple $M := (X, v, e, s)$ will act as the user's multi-coupon.

Redeem. In the redeem protocol, $\mathcal{U}$ (randomly) chooses an unspent coupon x_j from the tuple X, sets $x := x_j$ and $X' := X \setminus (x)$. The value x then becomes a common input to the redeem protocol. Next $\mathcal{U}$ proves to $\mathcal{V}$ that she is in possession of a valid multi-coupon ($\mathcal{V}$'s signature on X) containing x without revealing the signature itself.

In addition, we have the restriction that the index information of x, i.e. j, must not be disclosed to $\mathcal{V}$ when proving that x is part of the signed message in the CL signature.

If $\mathcal{V}$ would learn the index, he would be able to break the unlinkability property. To see this, simply suppose that two different coupons x and y are revealed both with respect to the base a_j from the CL signature. In this case, $\mathcal{V}$ immediately learns that the corresponding multi-coupons are different, since clearly $(z_1, z_2, \ldots, z_{j-1}, x, z_{j+1}, \ldots, z_m)$ $\neq (z'_1, z'_2, \ldots, z'_{j-1}, y, z'_{j+1}, \ldots, z'_m)$, where the z_i and z'_i are the other single coupons from the multi-coupon of x and y, respectively. So in fact, by revealing the index j, it is proven that x is the j-th component of an *ordered* set of messages $(x_1, x_2, \ldots, x_m)$, where the x_i, with $i \neq j$, are unknown to $\mathcal{V}$. However, in order to retain unlinkability the index must not be disclosed and we need to prove that x is contained in an *unordered* set of messages, i.e., $x \in \{x_1, x_2, \ldots, x_m\}$ where the x_i, $i \neq j$, are unknown to $\mathcal{V}$.

Note that proving that x is the j-th message of the signed message tuple X could be easily done by using Remark 1 from section 4.2: $\mathcal{V}$ computes a modified public key $PK_j := (A \backslash (a_j), b, c/a_j^x)$ and $\mathcal{U}$ runs the protocol $PoKSign$ with respect to PK_j, X', and (v, e, s). This way the signature (v, e, s) and X' would still be kept from $\mathcal{V}$, though the index j would be disclosed by the public key PK_j.

In order to overcome this index problem, $\mathcal{U}$ runs an extended version of the $PoKSign$ protocol from Figure 2 which proves that x is part of the signature but does not disclose the index of the spent coupon. The idea for this extension is as follows. Instead of disclosing to $\mathcal{V}$ which concrete public key PK_i $(i = 1, \ldots, m)$ is to be used for the verification, $\mathcal{U}$ proves that one of the public keys PK_i is the verification key with respect to the signature (v, e, s) on the message X'. For this, the proof PoK is extended with the $PoKOr$ protocol which adds the proof for the term $\bigvee_{i=1}^{m} C_i = T^{\epsilon} \prod_{l \in \{1,\ldots,m\}, l \neq i} a_l^{\xi_l} b^{\sigma}$ to PoK (see also Section 4.3). —Note that the terms $C_i = c/a_i^x$ are computed by both $\mathcal{U}$ and $\mathcal{V}$.— Using $PoKOr$, $\mathcal{U}$ proves that she knows the DLRep of one of the C_i with respect to its corresponding basis $(T, A \backslash (a_i), b)$ without revealing which one — since x is equal to x_j, the commitment $C_j = c/a_j^x$ must have a representation to the basis $(T, A \backslash (a_j), b)$ which is known to $\mathcal{U}$. Also note that this proves knowledge of the signature $(T = vb^w, e, s^* = s - ew)$ with respect to the public key $PK_j := (A \backslash (a_j), b, c/a_j^x)$. This is according to Remark 1 the same as proving it with respect to the public key (A, b, c) and, by Remark 2, the randomised signature (T, e, s^*) is, from $\mathcal{V}$'s point of view, equivalent to the signature (v, e, s). Hence, x must be part of a valid multi-coupon, i.e., a component from a message tuple that was signed by the vendor.

Eventually, if the proof protocol yields $accept$ then $\mathcal{V}$ is convinced that $\mathcal{U}$ owns a signature on an m-tuple X which contains x. At last $\mathcal{V}$ checks if x is already stored in his database of redeemed coupons. If it is not, $\mathcal{V}$ accepts x as a valid coupon and will grant the service.

5.1 Properties

In the following, we sketch how the coupon system proposed in the previous subsection satisfies the requirements from section 3.1. We will analyse the security of the system assuming that the strong RSA assumption holds.

Unforgeability. The property of unforgeability of our coupon system follows from the unforgeability of the CL signature scheme. As described in the previous section, a set of multi-coupons is a single CL signature on a block of messages. As has been proven in [9], forging CL signatures would break the strong RSA assumption.

Resetting the number of spent coupons requires to change some component in the tuple X, e.g., replacing a redeemed coupon x_i with $x_i^* \neq x_i$, since the vendor stores each spent single coupon x_i. However, replacing x_i by x_i^*, yielding tuple X^*, must be done such that $Sign_{(\cdot)}(X) = Sign_{(\cdot)}(X^*)$. Suppose the latter can be done. Then, we get $v^e \prod_{i=1}^{m} a_i^{x_i} b^s \equiv v^e \prod_{j=1}^{i-1} a_j^{x_j} a_i^{x_i^*} \prod_{j=i+1}^{m} a_j^{x_j} b^s$. Dividing by the right hand side yields $a_i^{x_i - x_i^*} \equiv 1 \bmod n$. Since $x_i \neq x_i^*$ it must be the case that $x_i - x_i^* = \alpha \cdot ord(\mathbb{Z}_n)$.

Now, choose any e such that $1 < e < (x_i - x_i^*)$ and $gcd(e, x_i - x_i^*) = 1$. By the extended Euclidean algorithm we can find d such that $ed + (x_i - x_i^*)t = 1$. Using this, we can compute e-th roots in $\mathbb{Z}_n$. For this, let u be any value from $\mathbb{Z}_n^*$ and compute $w := u^d$. Since $u \equiv u^{ed+(x_i - x_i^*)t} \equiv u^{ed} u^{\alpha \cdot ord(\mathbb{Z}_n) \cdot t} \equiv (u^d)^e \equiv w^e \bmod n$, the value w is an e-th root of u. This means we would have found a way to break the strong RSA assumption. Since this is assumed to be infeasible x_i cannot be replaced by $x_i^* \neq x_i$ without changing the signature (v, e, s).

Double-spending detection. If a cheating user tries to redeem an already spent single coupon x_i, she will be caught at the end of the redeem protocol, since the coupon to be redeemed must be disclosed and, thus, can easily be looked up in the vendor's database.

Redemption limitation. An m-redeemable coupon M cannot be redeemed more than m times (without the vendor's consent). Each multi-coupon M contains a signature on an m-tuple $(x_1, \ldots, x_m)$ of single coupons and in each run of the issue protocol a single coupon x_i is disclosed. Thus, after m honest runs using the same M, all x_i will be disclosed to the vendor. As argued under *unforgeability* and *double-spending detection*, already redeemed x_i cannot be replaced by fresh x_i^* and any attempt to 'reuse' an already disclosed x_i will be caught by the double-spending check.

Weak protection against splitting. Suppose that two users $\mathcal{U}_1$ and $\mathcal{U}_2$ want to share multi-coupon $M := (X, v, e, s)$ such that $\mathcal{U}_1$ receives single coupons $X_1 := \{x_1, \ldots, x_i\}$ and $\mathcal{U}_2$ receives the remaining coupons $X_2 := \{x_j, \ldots, x_m\}$, $i < j$. To achieve splitting, they have to find a way to make sure that $\mathcal{U}_1$ is able to redeem all $x_j \in X_1$ while not being able to redeem any coupon $x'_j \in X_2$, and analogously for $\mathcal{U}_2$. However, in the redeem protocol it is necessary to prove knowledge of the DLRep of C_X, which is X. Since proving knowledge of X while knowing only X_1 or X_2 would violate the soundness of the employed proof of knowledge $PoKRep$, and hence violate the strong RSA assumption, this is believed to be infeasible. Again, the missing part of X, either X_1 or X_2, cannot be replaced by 'fake' coupons $X'_{1|2}$ since this violates the unforgeability property of t he coupon system. Hence, X cannot be split and can only be shared if both $\mathcal{U}_1$ and $\mathcal{U}_2$ have full knowledge of X which comes down to *all-or-nothing sharing*.

Unlinkability. For unlinkability, we have to consider two cases, unlinkability between issue and redeem protocol runs and between executions of the issue protocol.

(1) *issue vs. redeem*: The issue protocol is identical to the protocol $BlindSign$ and, hence, the vendor $\mathcal{V}$, acting as signer, does not learn anything about the message X

being signed. However, $\mathcal{V}$ has partial knowledge of the signature (v, e, s), i.e., at the end of the issue protocol since he knows (v, e) but not s. To exploit this knowledge, he would have to recognize v or e in a run of the redeem protocol. In the redeem protocol, however, the user only sends commitments T and C_i $(i = 1, \ldots, m)$ to $\mathcal{V}$. Since the commitment scheme is statistically hiding the vendor cannot learn anything about v or e from the commitments.

Furthermore, all sub-protocols in $Redeem$ operate only on the commitments T and C_i and since the opening information of either T or any C_i is never given to $\mathcal{V}$ during the execution of any of these proof protocols the vendor is unable to recognise v or e.

(2) *redeem vs. redeem*: The redeem protocol mainly consists of the $PoKSign$ protocol which only employs zero-knowledge proofs and statistically hiding commitments and, hence, is unlinkable. However, in the redeem protocol the coupon value x is given to the vendor $\mathcal{V}$, i.e., the verifier. In the following we sketch that $\mathcal{V}$ cannot use this information to infer other information that helps him to link redemptions of single coupons?

To see this, let τ be the transcript of the redeem protocol where x is released. Since all proofs of knowledge applied in the redeem protocol perform challenge-response protocols, there will be some values u, containing x, which were formed by the user in response to challenges t chosen by $\mathcal{V}$. The general form of a response is $u = ty + r$, where t is $\mathcal{V}$'s challenge, y is some committed value of which knowledge is proven, and r is a witness randomly chosen by the user. However, since x's index is not revealed (due to $PoKOr$) every response u_i $(i = 1, \ldots, m)$ is equally likely to contain x.

Now, if $\mathcal{V}$ guesses x's index, say j, he would only learn the value r_j from the response u_j. However, this reveals no information of any other response u_i, $i \neq j$, from τ, since for any value x_i, contained in the response u_i, the witness r_i is randomly and uniformly chosen anew for each u_i. Hence, from $\mathcal{V}$'s point of view u_i is a random value and may contain any value x_i' and, thus, x_i is (still) statistically hidden in u_i.

The consequence of the arguments mentioned above is that given any two redeem protocol transcripts $\tau^{(x)}$ and $\tau^{(y)}$ where x and y are deducted from (hidden) multi-coupons $M^{(x)}$ and $N^{(y)}$, respectively, the verifier cannot determine whether x is hidden in any response $u_i^{(y)}$ from $\tau^{(y)}$ and analogously if y is hidden in any $u_l^{(x)}$ from $\tau^{(x)}$. This means the vendor cannot decide with a non-negligible probability better than pure guessing if the multi-coupons $M^{(x)}$ and $N^{(y)}$ are related or unrelated.

Minimum Disclosure. A further consequence of the unlinkability of transactions in the coupon system, and due to the fact that no counter value is sent in any protocol, the number of unspent coupons cannot be inferred from any redeem protocol run.

6 Conclusion

The coupon system presented in this work allows vendors to issue multi-coupons to their customers, where each single coupon of such a multi-coupon can be redeemed at the vendor's in exchange for some good, e.g., an MP3 file, or some service, e.g., access to commercial online articles of a newspaper. Issuing coupons is advantageous to vendors since coupons effectively retain customers as long as they have coupons left to spent. However, multi-coupons might be misused by vendors to link transactions of customers in order to collect and compile information from their transactions in a profile. To protect the privacy of customers in this respect, the coupon system that we proposed allows customers to unlinkably redeem single coupons while preserving security requirements of vendors.

Acknowledgement

The authors are grateful to the anonymous reviewers who provided valuable comments which greatly helped to improve the quality of this paper.

References

1. F. Boudot. Efficient proofs that a commited number lies in an interval. *Adv. in Cryptology - EUROCRYPT 2000*, LNCS 1807. Springer, 2000.
2. S. Brands. An efficient off-line electronic cash system based on the representation problem. CWI Report, CS-R9323, Centrum voor Wiskunde en Informatica (CWI), 1993.
3. S. Brands. *Rethinking Public Key Infrastructure and Digital Certificates – Building in Privacy*. PhD thesis, Eindhoven Institute of Technology, The Netherlands, 1999.
4. G. Brassard, D. Chaum, C. Crepéau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37, 1988.
5. E. Brickell, J. Camenisch, L. Chen. Direct anonymous attestation. *Proc. 11th ACM Conference on Computer and Communications Security*, pages 132-145, ACM press, 2004.
6. J. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, Switzerland, 1998.
7. J. Camenisch, J. Groth. Group signatures: better efficiency and new theoretical aspects. *Forth Int. Conf. on Security in Communication Networks – SCN 2004*, LNCS 3352, Springer, 2005.
8. J. Camenisch, A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *EUROCRYPT '01*, LNCS 2045. Springer, 2001.
9. J. Camenisch, A. Lysyanskaya. A signature scheme with efficient protocols. *Third Conference on Security in Communication Networks – SCN'02*, LNCS 2576. Springer, 2002.
10. J. Camenisch, M. Michels. Separability and efficiency for generic group signature schemes. *Adv. in Cryptology - CRYPTO '99*, LNCS 1666. Springer Verlag, 1999.
11. D. Chaum. Privacy protected payments: Unconditional payer and/or payee untraceability. *Smart Card 2000, Proceedings*. North Holland, 1989.
12. L. Chen. Access with pseudonyms. *Cryptography: Policy and Algorithms, International Conference, Brisbane, Australia, July, 1995, Proceedings*, LNCS 1029. Springer, 1996.
13. R. Cramer, I. Damgård, B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *Adv. in Cryptology - CRYPTO '94*, LNCS 839. Springer, 1994.
14. I. Damgård, E. Fujisaki. A statistically hiding integer commitment scheme based on groups with hidden order. *Adv. in Cryptology - ASIACRYPT '02*, LNCS 2501. Springer, 2002.
15. M. Enzmann, M. Fischlin, M. Schneider. A privacy-friendly loyalty system based on discrete logarithms over elliptic curves. *Financial Cryptography*, LNCS 3110, Feb. 2004.
16. F. Feinberg, A. Krishna, Z. Zhang. Do we care what others get? A behaviorist approach to targeted promotions. *Journal of Marketing Research*, 39(3), Aug. 2002.
17. N. Ferguson. Extensions of single term coins. *Adv. in Cryptology - CRYPTO '93*, LNCS 773. Springer, 1993.
18. E. Fujisaki, T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. *Adv. in Cryptology - CRYPTO '97*, LNCS 1294. Springer, 1997.

19. D. Hoffman, T. Novak, M. Peralta. Building consumer trust online. *Communications of the ACM*, 42(4), Apr. 1999.
20. A. Kobsa. Tailoring privacy to users's needs. *User Modeling 2001 (UM 2001)*, LNAI 2109. Springer, 2001.
21. G. Macintosh, L. Lockshin. Retail relationships and store loyalty: A multi-level perspective. *International Journal of Research in Marketing*, 14(5), Dec. 1997.
22. D. Maher. A platform for privately defined currencies, loyalty credits, and play money. *Financial Cryptography*, LNCS 1465. Springer, 1998.
23. G. O'Connor, R. O'Keefe. The Internet as a new marketplace: Implications for consumer behaviour and marketing management. *Handbook on Electronic Commerce*. Springer, 2000.
24. A. Odlyzko. Privacy, Economics, and Price Discrimination on the Internet. *5th International Conference on Electronic Commerce (ICEC 2003)*. ACM Press, 2003.
25. T. Okamoto, K. Ohta. Disposable zero-knowledge authentications and their applications to untraceable electronic cash. *Adv. in Cryptology - CRYPTO '89*, LNCS 435. Springer, 1990.
26. R. Pear. Electronic cards replace coupons for food stamps. New York Times, June 23, 2004.
27. P. Persiano, I. Visconti. An efficient and usable multi-show non-transferable anonymous credential system. *Financial Cryptography*, LNCS 3110. Springer, Feb. 2004.
28. A. de Santis, G. di Crescenzo, G. Persiano, M. Yung. On monotone formula closure of SZK. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, November 1994.
29. G. Shaffer, Z. Zhang. Competitivec coupon marketing. *Marketing Science*, 14(4), 1995.
30. C. Shapiro, H. Varian. *Information Rules*. Harvard Business School Press, 1999.
31. E. Verheul. Self-blindable credential certificates from the weil pairing. *Adv. in Cryptology - ASIACRYPT '01*, LNCS 2248. Springer-Verlag, 2001.
32. A. Wibowo, K. Lam, G. Tan. Loyalty program scheme for anonymous payment systems. *Electronic Commerce and Web Technologies*, LNCS 1875. Springer, 2000.