An Architecture for Secure Generation and Verification of Electronic Coupons

Rahul Garg Parul Mittal Vikas Agarwal
Natwar Modani
{*grahul, mparul, avikas, mnatwar*}*@in.ibm.com*
IBM India Research Lab.

Abstract

Coupons are very useful mechanism for carrying out different marketing management functions like sales promotion, brand promotion, and inventory management. With the advent of Internet shopping and online stores, there is an immediate need for an electronic equivalent of traditional paper coupons. Security issues such as coupon tampering, exchange, duplication and double spending become very important for electronic coupons. Although the security issues in electronic coupons appear to be similar to those in electronic cash systems, there are significant differences that require the design of a different protocol to carry out secure coupon transactions.

In this paper we describe a system for secure generation and verification of electronic manufacturer and store coupons. The proposed solution is based on a third party centralized coupon mint which carries out the check for double spending, similar to online electronic cash systems. However, unlike electronic cash systems, the coupon mint remains completely unaware of the promotion details (the amount of discount, product details etc.) and simply provides an infrastructure for online coupon verification. Thus, the coupon mint service can be provided by semi-trusted third parties different from manufacturers. The proposed system is inherently distributed and scalable. It lets different manufacturers independently choose their own promotion and targeting policies (without involving the coupon mint) and the coupon mint service provider.

The system also offers several new types of coupons like aging coupons, growing coupons, random value coupons, and early bird coupons which were not practical by using traditional paper coupons (and not possible by using the electronic cash protocols).

1 Introduction

Coupons have been traditionally used to carry out different marketing management functions like sales promotion, brand promotion, and inventory management [1]. As a result of advancement in the Internet and e-commerce technologies, a large number of online stores offering a variety of products and services have been opened. These electronic stores also like to issue electronic coupons to their customers for sales promotion and other marketing management functions.

There can be different types of coupons depending on the issuing authority (manufacturer or store) [2], whether they are targeted for a set of potential customers or they are untargeted, whether their distribution is limited or not, and whether the value of the offer is fixed and known in advance to the customer. Promotions are designed using these coupon types to achieve a specific marketing objective. For instance, a manufacturer may issue some targeted coupons for limited distribution for product trials and a store may issue coupons for unlimited distribution to get rid of unsold inventory of perishable items.

The online nature of electronic coupons makes them prone to various kinds of frauds. A number of manufacturers are not adopting Internet coupons because of security concerns [3]. As a result, the security issues [3] have become even more important for electronic coupons. There is an immediate need for a generic electronic coupon system that is secure, can be trusted and can offer a wide variety of coupon types including those which are currently in practice.

Double spending is the most difficult security problem to handle in case of manufacturer electronic coupons. Several solutions to this problem have been proposed in the context of electronic cash [4, 5, 6]. However, none of the proposed solutions are applicable for electronic coupons because of the differences in the way electronic cash and electronic coupons are used. A bank may be willing to provide a highly available online verification infrastructure for its business, whereas a manufacturer may not be willing to provide the same for its electronic coupons. If double spending is detected at a later stage, banks may be able to recover the doubly spent money from the customer responsible for it (through debit or litigation), whereas a manufacturer may not be able (and willing) to do so for the doubly redeemed electronic coupons. Customers may be willing to carry a special card for their electronic cash, but not for electronic coupons.

Currently, there are a number of Internet web sites that offer coupons online, as an image or a bar code, that the user can print on a local printer and use in a particular physical store. Such coupons are not really electronic coupons as they can only be used at a physical store. They are best described as traditional paper coupons distributed on the Internet.

Online stores on the Internet have also started issuing coupons. Most of the current offerings give coupons that can only be used at the issuing store and are very limited. Kumar et al. [2] have proposed an electronic coupon architecture which is limited to single store electronic coupons. Currently, there is no secure system for generic electronic manufacturer coupons.

Other related problems are that of "hit-shaving" [7] and "hit-inflation" [8]. Hit shaving occurs when a target web site (advertiser) omits the references of some of the hits to its web site. Hit inflation occurs when a referrer's site (say a portal) artificially generates hits to the advertiser's site that do not correspond to a genuine user's visit to the site. These problems have become important especially in the context of advertisement on the Internet that use a click-through payment program. A detailed study on these can be found in Reiter et al. [7] and Anupam et al. [8].

In this paper we describe an architecture for secure generation and verification of electronic coupons of different types. The proposed solution is based on a third party centralized coupon mint which carries out the check for double spending. However, unlike electronic cash systems, the coupon mint remains completely unaware of the promotion details (the amount of discount, product details etc.) and simply provides an infrastructure for online coupon verification. The system also offers several new types of coupons like aging coupons, growing coupons, random value coupons, and early bird coupons, which were not possible by using traditional paper coupons (and the electronic cash protocols).

We begin by describing the different types of paper coupons that are currently in practice in Section 2. We describe newer types of electronic coupons including aging coupons, growing coupons, random value coupons, early bird coupons that can be provided by our architecture. We also discuss the important security issues in a generic electronic coupon system and describe why the solutions proposed for electronic cash systems are not directly usable in an electronic coupon system. We describe our architecture and protocol in detail in Section 3. We also discuss how different types of frauds can be prevented by the proposed architecture. We discuss some implementation issues in Section 4 and we describe our implementation in brief in Section 5. We present the performance of our prototype implementation in Section 6 and conclude in Section 7.

2 Electronic Coupons: Possibilities and Pitfalls

There are several types of coupons. In this section, we first introduce the different types of coupons and then discuss security issues for a generic online electronic coupon system. We also discuss why protocols proposed for electronic cash systems cannot be used in their present form in an electronic coupon system.

2.1 Coupon Classification

Manufacturer vs. Store coupons. This classification is based on the coupon issuing authority. A store may distribute discount coupons on a few products to attract customers to the store. These coupons can only be redeemed at the particular store. The cost of such a promotion is borne completely by the store issuing the coupons. After collecting some relevant information (needed for future profiling, evaluating the scheme etc.), the redeemed coupons may be discarded by the store. Such coupons are called store coupons. On the other hand, a leading brand manufacturer may periodically issue discount coupons to its loyal customers. The customer may take these coupons to any store for redemption. Such coupons are called manufacturer coupons. The cost of such a promotion is borne entirely by the manufacturer. The stores claim the discount given to the customer by sending all the redeemed coupons to their respective manufacturers. This process is called coupon clearing. The manufacturer uses the cleared coupons to gather important sales and redemption information. In addition to the discount amount, the manufacturer may also pay a handling fee to the stores for each manufacturer coupon redeemed at these. There may be coupons that fall between manufacturer coupons and store coupons, such as manufacturer coupons that can be redeemed at selected stores, or coupons that can be redeemed on any store of a retail chain, or coupons valid only at stores participating in a given program.

Targeted vs. Untargeted coupons. Many times manufacturers or stores want to issue certain coupons to only a selected group of customers. For instance, a leading brand may wish to issue coupons targeted to the regular customers of a competitor brand, to induce brand switching. A targeted coupon is intended for a particular customer (or a set of customers), whereas an untargeted coupon may be used by anyone.

Limited distribution vs. Unlimited distribution coupons. The coupon issuers often want to control the number of coupons of a particular type that are distributed. By limiting the distribution, the issuer can limit the amount of discount to be given, and hence estimate the overall cost of the promotion (say in a product trial promotion). Such coupons are called limited distribution coupons. Sometimes manufacturers or stores prefer to distribute a large number of discount coupons instead of a price mark-down. Such coupons are called unlimited distribution coupons.

Variable value coupons. Finally, yet another classification of coupons is based on the coupon value. Most of the printed coupons have fixed value known in advance. However, with the proposed electronic coupon system, it is possible that the coupon value is determined dynamically, based on certain parameters. This results in a number of exciting possibilities, such as gaming, lottery etc., which may be used for defining more effective promotions. An *early bird coupon* is one where the first "k" customers who bring the coupon to a store get the discount. An *aging coupon* is one whose value decays with time. A *growing coupon* is one whose value increases with time. A *random value coupon* is the one whose value is not known at the time of issuing. The value of such a coupon is known only after a purchase is made. However, the range and the probability distribution of the possible values may be known in advance. A lottery ticket is an extreme example of a random value coupon, where the coupon value is either zero or a large sum (the lottery prize) and the statistical distribution of possible coupon values is known in advance. The coupon value is known only after the purchase (of lottery ticket) is made.

2.2 Security Issues

Tampering and double spending are two important security issues in an online currency of any form. In the context of electronic coupons, a customer may change the discount amount or other terms (such as validity period) of an electronic coupon to get an illegitimate discount. In the case of manufacturer coupons, even the retailers may alter the coupon before sending it for clearing. The problem of tampering is usually solved by using digital signatures [9, 10].

The problem of double spending becomes severe especially in the case of manufacturer coupons. Since electronic coupons can be duplicated easily, a customer may use the same manufacturer coupon at two different retailers. The manufacturer will not know this until both retailers send their coupons for clearing. In a more contrived scenario, a group of retailers may duplicate and share the manufacturer coupons redeemed at their stores and blame customers for double spending.

There are online as well as off-line solutions for the double spending problem in the electronic cash literature. The off-line solutions that prevent double spending require the users to store the cash in a special *tamper-proof* [11] hardware. Such a solution may be difficult to use for electronic coupons as users may not be willing to get a special hardware just for storing their electronic coupons. There is another class of off-line solutions [4, 5, 6] that do not prevent double spending but detect the identity of user involved in such a fraud. Double spending is detected only when two or more copies of the money spent twice is deposited back into the bank. While such off-line solutions may work for electronic cash, they may not work for electronic coupons because of the nature of relationship between customers and the manufacturers. Customers do not have accounts with manufacturers and the manufacturers may not be able to charge (or sue) customers even if double spending is detected at a later stage.

The online solutions proposed for preventing double spending are similar to a system where the coupon clearing is done online at the time of purchase. In this case, if a coupon is redeemed for the second time, the manufacturer would inform the retailer and the retailer would refuse to accept the coupon. To implement an online coupon clearing system, the manufacturer will need to implement a highly reliable and available coupon clearing infrastructure, which is likely to fall beyond the abilities and interests of most manufacturers. The manufacturers may not be willing to completely trust third parties to provide this service to them, especially because it implies giving them access to their valued customer and sales data. Therefore, an approach is needed where retailers can verify the validity of coupons without requiring any online support from manufacturers.

Another important issue for electronic coupons is to prevent coupon trading among customers (unlike electronic cash systems where the goal is the facilitate the cash exchange among users). The purpose of targeting is defeated if the customers are able to freely exchange or transfer their coupons, possibly in return for money or other coupons. There are already several Internet web sites for coupon trading. In addition, the manufacturers would like to track the customers and their coupon usage patterns (unlike electronic cash systems where banks will like to provide anonymity to their customers) for evaluating the success of a promotion and for future targeting and profiling.

The manufacturer should be able to relate the coupons redeemed by a customer at a retailer to the sales of intended products. If this is not done, then the retailers may just collect the coupons and send them to manufacturers for clearing. The manufacturers should also be able to check if the retailers are properly checking all the terms and conditions of the coupons.

Finally, in the case of variable value coupons, the coupon value is not fixed and depends on time at which the coupon is being redeemed, the number of coupons of that promotion which have been redeemed earlier, and a random element. In this case, the retailer should be able to determine the correct coupon value without any online support from the manufacturer. In addition, the customer should be able to check that the coupon value determined by the retailer is indeed according to the parameters specified in the coupon. Finally, the manufacturer should be able to check that the coupon value reported in the coupon sent for clearing is indeed the correct coupon value.

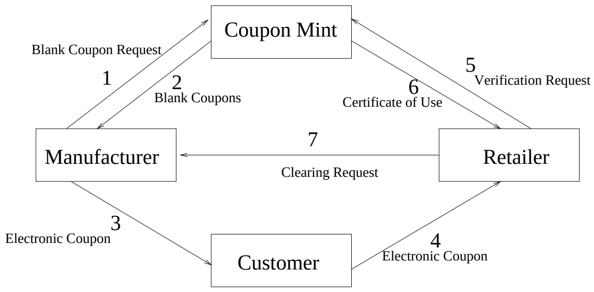

Figure 1: Components in the Electronic Coupon System.

In the following section, we describe an architecture for a coupon system which addresses all the security issues discussed above. The architecture is based on a third party *semi-trusted* centralized *coupon mint* which checks for double spending and helps in determining the fair coupon value, without knowing any other details about the coupon.

3 System Description

Figure 1 shows the main entities in a generic electronic coupon system. It consists of a coupon mint, a manufacturer, a customer and a retailer. A coupon mint is an independent third party service provider, which provides highly available online electronic coupon verification service to retailers on behalf of multiple manufacturers. The manufacturer represents the authority issuing coupons to promote its products or services. The customer represents the users to whom the coupons are issued and who make purchases of products or services from the retailers. The retailer represents an online store selling products and/or services.

```
<blankCouponRequest>  ::= <numCoupons> [<classId>] [<validity>] [<manufId>]
<blankCoupon>         ::= <couponId> <blankCouponExpiry> [<timeCreation>] [<classId>]
                          [<manufId>] [<verificationUrl>] <signature>
<electronicCoupon>    ::= <blankCoupon> <issuingAuthority> <discount> [<conditions>]
                          [<visual>] <signature>
<conditions>          ::= [<validityPeriod>] [<purchaseConditions>]
                          [<personalizationConditions>] [<otherConditions>]
<verificationRequest> ::= <blankCoupon> <invoiceNo> [<retailerInfo>]
<certificateOfUse>    ::= <blankCoupon> <invoiceNo> [<retailerInfo>] <numUsed>
                          [<numClassUsed>] <rand> <dateTime> <signature>
<clearingRequest>     ::= <electronicCoupon> <certificateOfUse>
```

Figure 2: Messages exchanged in the electronic coupon system. The terms enclosed in [ ] are optional.

Figure 2 shows the details of messages exchanged between these entities during the life cycle of an electronic coupon. All the messages are encrypted to ensure privacy.

In the following subsections we explain our electronic coupon protocol and the steps taken by each of the entities involved during coupon issue, redemption and clearing to safeguard against potential fraud.

3.1 Coupon Issuing

When a manufacturer needs to issue a coupon, it sends a *blank coupon request* to the coupon mint which then issues unique unforgeable blank coupons to the manufacturer. Unforgeability implies that when these coupons are presented to the coupon mint at a later stage: (a) it can recognize that these coupons have been generated by the coupon mint and (b) no other entity can produce a blank coupon which, with significant probability, may be recognized by the coupon mint as valid. Unforgeable coupons can be produced by using a long coupon identifier x and applying a keyed hash function [12] $f_k(x)$ to it, where k is the key known only to the coupon mint. The pair $(x, f_k(x))$ forms an unforgeable blank coupon. We use digital signatures, which are a special case of keyed hash functions, to generate unforgeable coupons as seen from the format of `blankCoupon` message in Figure 2.

The manufacturer then writes the coupon details on a blank coupon and digitally signs it to make it an authentic electronic coupon. Coupon details contain the discount amount, purchase conditions, validity period, personalization condition, other conditions and a human-readable description of the offer (such as an image or HTML text).

In the simplest case, the discount amount is simply a number. However, for variable value coupons, the discount amount is represented as a function which takes several parameters as input to compute the exact discount amount.

Validity period is represented as a start and end timestamp. The coupon is valid only in this period. In addition, a coupon may contain conditions indicating its validity during specified intervals of a day (lunch coupons), on specified days of a week (Sunday coupons), and in specified weeks of a month.

Purchase conditions represent the set of purchases required by a customer to obtain the discount. These may be a list of products (with given minimum units of necessary purchases), or products from a product family, or purchase totaling more than a given amount. For example, a *buy one and get one free* coupon will require the purchase of two units of a product and offer 100% discount on one unit of the product.

Personalization conditions ensure that only the targeted customers are able to redeem coupons. The simplest method to identify a customer is by a credit card number. Credit card based personalization conditions contain a one way hash function [12] of the targeted customer's credit card number, which is verified against the credit card number given at the time of online purchase. A one-way hash function ensures that the credit card number is not misused. These coupons may impose the restriction that the customer should use the same credit card for purchase with which the coupon is personalized. Such coupons are difficult to issue and distribute as the customer's credit card number (or its one-way hash) may not be known at the time of coupon distribution.

Other personalization conditions may be based on customer's email address, IP address, Internet service provider (ISP), profile based on a cookie, membership number, customer name, address and zip code. It is easier for online sites to know a customer's name or email address. Therefore, personalization based on name and email address is easier to carry out than credit card based personalization. However it may be relatively difficult to verify. In this case the retailer may require the customer to register at its site (and supply personal details such as name, email address etc.) before the customer can redeem coupons. It is also possible to carry out reasonable targeting using a combination of the above personalization conditions.

A coupon may contain some other conditions for its validity. Some coupons may be valid at selected retailers, some may be valid only if a given payment method is used.

3.2 Redemption

When the customer presents a coupon to a retailer the retailer first checks the integrity of coupon by verifying the manufacturer's digital signature. The retailer knows the list of products it sells and hence the list of manufacturers which may potentially issue coupons. Therefore it does not need a public key infrastructure [13] for obtaining public keys. It may periodically get public keys from its manufacturers.

Secondly, the retailer checks if the purchase conditions, validity period, personalization condition, and other conditions mentioned in the coupon are valid at the time of redemption.

Finally, if all the conditions are met, the retailer checks if the coupon has been redeemed earlier by the customer by sending a *verification request* to the coupon mint. The verification request consists of the blank coupon part contained in the electronic coupon and an invoice number identifying the current sale transaction. The coupon mint responds to verification request with a digitally signed *certificate of use* which indicates the number of times (numUsed) a verification request for the given coupon has been sent by the retailers (and hence the number of times the coupon has been redeemed earlier). The retailer uses this information to check for double spending. Note that since all the messages are encrypted, only a manufacturer or a customer can get access to the blank coupon part of her coupons. Since blank coupons are unforgeable, no entity other than the customer herself can send fake verification requests to invalidate other customer's coupons.

For early bird coupons, the retailer needs to know the number of coupons of the same type redeemed so far. For this, each early bird coupon of a promotion scheme is tagged with a class identifier (classId), which identifies the promotion. The coupon mint keeps track of the number of coupons of each class redeemed so far and reports this number in the certificate of use (numClassUsed). The early bird coupons also contain an early bird condition which specifies the maximum permissible value of numClassUsed.

Variable value coupons are implemented by specifying a discount function in place of the discount amount. This function has four input parameters: date and time-stamp (dateTime), number of times the coupon has been redeemed earlier (numUsed), number of coupons of this coupon class redeemed so far (numClassUsed) and a random number (rand). The certificate of use contains all the input parameters of this discount value function. The retailer applies this function to the certificate of use and calculates the discount amount. Neither the retailer nor the customer has control over certificate of use. So neither can manipulate the discount amount to their advantage. The coupon mint has no incentive to give an incorrect certificate of use. The only possibility of fraud is when the retailer and the coupon mint (and optionally the customer) collude and design a certificate of use which computes the discount amount to their advantage.

A number of functions may be designed by the manufacturer for early bird, aging, growing, and random coupons. For example, for aging and growing coupons the function value depends on dateTime, for early bird coupons the function value depends on numClassUsed and for random value coupons the function value depends on rand. A number of interesting combinations of these coupon types are also possible.

3.3 Clearing

At a later stage, the retailer sends all the redeemed coupons along-with their certificate of use to their respective manufacturers for clearing. The manufacturer checks for integrity of electronic coupons and certificate of use by verifying the digital signatures. The certificate of use acts as a proof for the retailer that the coupon was redeemed at its store. For variable value coupons, the manufacturer checks using the function specified in the coupon, and its certificate of use, that the discount value was determined fairly by the retailer. If all these conditions are met, the manufacturer sends the required amount of money to the retailer.

The certificate of use also contains the retailer invoice number and optional sales related information (`retailerInfo`), which is used by the manufacturer to correlate a coupon redemption with a product purchase at the retailer. The manufacturer may also carry out periodic audits at retailers to check if the retailers are properly verifying all the conditions mentioned in the coupon.

It is to be noted that in the end, the manufacturers also get all the information about the usage pattern of redeemed coupons which can be used for designing future promotions. Also in the entire process, the coupon details are never revealed to the coupon mint. The manufacturers need to trust the coupon mint only for variable value coupons. If the coupon mint issues a valid certificate of use for a coupon that has been redeemed earlier, the manufacturer can detect it as it will finally get two valid certificate of use from two different retailers.

4 Discussion

The system described in Section 3 is a generic and secure system for electronic coupon generation and verification. It solves the problem of coupon forging, tampering, double spending, unwanted coupon exchanges, fair determination of coupon value for variable value coupons and ties coupon redemption to product purchase.

Separation of the online coupon verification functionality from coupon issue and distribution is the main contribution of this architecture. Online coupon verification requires highly available online infrastructure whereas designing a promotion scheme, targeting, and coupon distribution requires data mining and domain-specific marketing knowledge. By separating the coupon verification and distribution, different manufacturers and stores can use their own promotion and coupon distribution policies, without maintaining online verification infrastructure of their own.

The architecture is generic enough to support manufacturer, store and group-of-store coupons for online stores. In the case of store coupons, the functionalities of manufacturer and retailer are co-located at the store. For group-of-stores coupons, the "other conditions" in the coupon contain a list of member stores where the coupon may be redeemed.

All verification requests coming to a centralized server raises serious scalability concerns. However it is easy to see that there may be several such coupon mints providing the verification service. Thus a manufacturer has a choice of selecting a coupon verification provider. It may also dynamically switch to a new verification provider, in case it finds the service of one unacceptable. The manufacturer writes a verification URL in the coupon which the retailer follows in order to send verification request. The coupon mint may also put a verification URL in the blank coupon to distribute its verification load across multiple verification servers.

The coupon mint needs to store information about the coupons redeemed only for a limited period. Each blank coupon has an expiry date. The validity period of a coupon must entirely be contained within this expiry date. The coupon mint keeps a record for each redeemed coupon only till its blank coupon expiry. Thus, if a verification request for an already redeemed coupon comes after its blank coupon expiry, the coupon mint simply rejects the request on the basis of expiry.

5 Implementation

The system software comprises of three main components, one each for the coupon mint, the manufacturer and the retailer. Optionally a fourth component, for storage of customer's electronic coupons, is also provided. Each component is a stand alone Web based entity implemented as a Java servlet.

The cryptographic operations are written in the C language using Java Native Interface (JNI) for performance reasons. In order to adhere to standards, we used XML [14] to encode all the messages and SSL (TLS) [15][1] to encrypt them.

5.1 Coupon Mint Component

The coupon mint component receives blank coupon requests from the manufacturers and verification requests from the retailers as HTTP/POST [16] messages and sends blank coupons or certificates of use as an XML document in responses.

5.2 Manufacturer Component

The manufacturer component allows multiple manufacturers to define different promotions, through a Web based interface. For each promotion, the manufacturer supplies the promotion details, such as the number of coupons to be distributed, the discount amount, the product ID etc. The software obtains an appropriate number of blank coupons from the coupon mint, writes promotion specific information on them and stores them.

The manufacturer component also provides an HTTP interface to distribute the coupons. This interface may be used to flash electronic coupons as banner advertisements on various Internet web sites. The manufacturer component gets a message to serve an electronic coupon advertisement. With the message, it gets the customer profile and personalization information. It shows a coupon image targeted to the customer. Clicking on a coupon image generates an HTTP/GET message at the manufacturer. The manufacturer component then writes the personalization information on the coupon, digitally signs it and, depending on the interface, either saves it on the customer's hard disk or uploads it to a third party *coupon storage service provider.*

5.3 Customer Component

A customer typically runs a web browser on a desktop to carry out on-line purchases. Any additional software on the customer machine may discourage the customer from using electronic coupons. Besides, any additional customer side software also restricts the customer to use the same machine every time she wants to use the electronic coupons. Therefore we avoided any additional software on the customer side. Since the customer uses a web browser, customer-retailer interface and the customer-manufacturer interface is based on the HTTP protocol.

[1] TLS is the IETF proposed standard for SSL. The SSL protocol version 3.0 is available as an Internet draft.

5.4 Retailer Component

The retailer component runs at various online stores. This component has to be integrated with the e-commerce software at each online store. The integration is done with the order processing module of the online store's e-commerce server. It is expected that use of Java to code this component would facilitate integration with different e-commerce servers.

Once a customer decides to buy some items from an online store and proceeds to process the order, the customer is given the option to redeem coupons. The retailer component provides an interface to upload electronic coupons from a customer's desktop or coupon storage provider, checks the coupons for applicability to the current purchase order, verifies the applicable coupons as described in Section 3, and finally reports the discount amount to the commerce server.

The uploading of coupons from the customer's desktop uses the "File Upload" proposed enhancement to HTML [17]. The interface with the coupon storage service provider comprises of a HTTP/POST request, sent to the coupon storage service provider to get the applicable coupons. The request contains the information about the items being purchased. The response from the coupon storage service provider contains the customer's coupons in XML, whose purchase conditions are satisfied by the items being purchased. However, such an interface may require a significant standardization effort. In general, there may be several purchase conditions in a coupon making some of the coupons mutually exclusive. There can be scenarios when there are multiple mutually exclusive coupons that can be used for the set of purchased items. The retailer component checks such situations and asks the customer to select a subset of non-mutually exclusive coupons.

An advanced decision support system to help the customer choose the best coupon subset, for a given purchase order, may be implemented in future.

In order to check the validity of selected coupon subset, the retailer component checks the manufacturer's signature on the coupons using manufacturer's public key. It re-checks the purchase conditions and checks the validity, personalization and other conditions. For this, the retailer component needs purchase order details and customer specific information (such as customer name, membership number, credit card number etc.) from the e-commerce server. Interfaces based on the purchase order identifier are defined, using which the retailer component can get customer-specific information from the e-commerce server for a given order. It is assumed that every purchase order in an e-commerce server will have a unique invoice number which is passed in the HTTP/POST messages to the retailer component.

Finally, for checking double spending and obtaining the certificate of use, the retailer component sends a HTTP/POST request containing the blank coupon part of the coupon, invoice number, and optional information. The coupon mint returns a certificate of use in XML format. The retailer component stores the coupon and the corresponding certificate of use for clearing with the manufacturer. The retailer component then computes the discount amount and returns it to the e-commerce server using a predefined API.

6 Performance Measurements

We measured the performance of our prototype implementation to estimate the resources required for a production system.

6.1 Experimental Setup

The testbed consists of one server and one client system. The server system is a 600 MHz Pentium-III with 128MB of RAM, and a 100 Mbps Ethernet network interface running RedHat Linux 6.2. The client system is a 233MHz Pentium-II with 96MB of RAM, running RedHat Linux 6.0. The client communicates with the server through a 100 Mbps LAN.

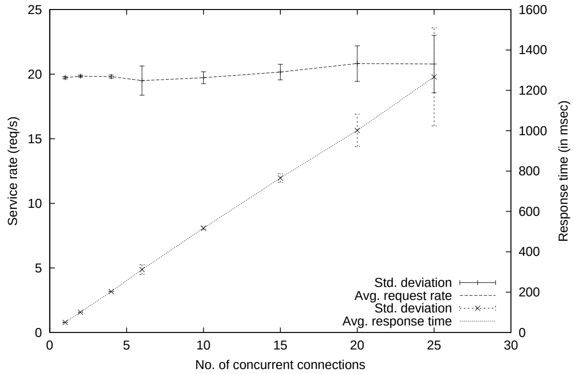

Figure 3: Peak performance: Blank coupon request.

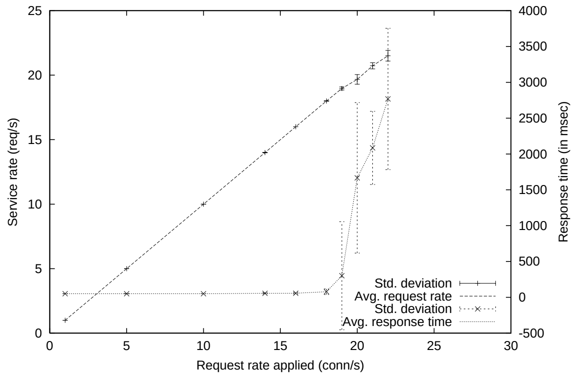

Figure 4: Typical performance: Blank coupon request.

The server system runs an Apache web server (version 1.3.12) with dynamically linked Apache_Jserv module (version 1.1.2) to process HTTP requests that invoke various Java servlets. We used Sun JDK 1.2.2 with user-level threads and no just-in-time (JIT) compilation. It runs a MySQL database server (version 3.23.27) for serving database requests. It also runs the coupon mint component, the manufacturer component, the coupon storage provider component and the retailer component. We used the cryptlib encryption toolkit[2] for performing cryptographic operations in C.

The client system runs httperf [18], a widely used Web performance measurement tool. Since httperf does not support SSL, it was disabled during the measurements. To test the system performance, Apache web server was configured to support persistent connections with infinite number of requests on each persistent connection, with number of start

[2] http://www.cs.auckland.ac.nz/~pgut001/cryptlib/

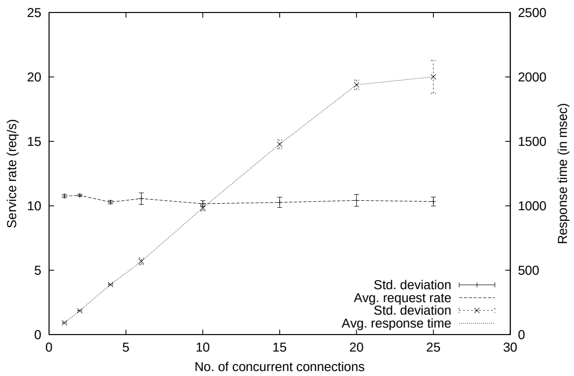

Figure 5: Peak performance: Verification request.

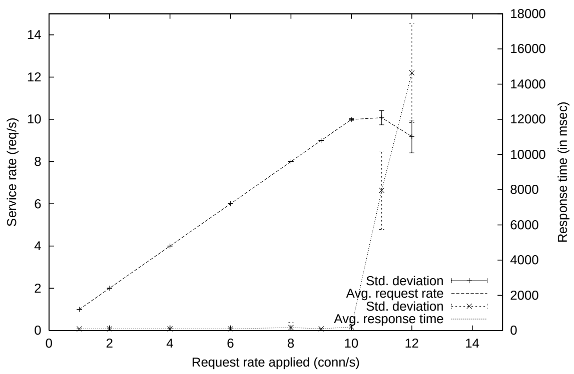

Figure 6: Typical performance: Verification request.

servers as 20 and the maximum number of simultaneous connections as 20. This is to limit the number of Apache processes as there is one process per active connection. The Apache_Jserv module was configured to disable authentication of the requesting client, auto-reloading of classes upon modification and logging except for error messages. These are factors affecting Apache_Jserv performance. To optimize database access time, a set of database connections were opened at servlet initialization. Successive requests obtain a free connection from the pool of database connections and return it to the pool, after the request is processed.

The tests were conducted for main functions of the coupon mint, namely the blank coupon generation and verification request.

For each of the messages, httperf sends appropriate number of HTTP requests so that it runs for approximately 150 seconds. The following tests were conducted for each of the messages.

Peak performance: To determine the peak performance of the server. In this test, the client repeatedly sends HTTP requests, one request at a time, on one or more persistent HTTP/1.1 connections(s). The number of simultaneous persistent connections is increased, starting from 1, till the server saturates. The requests are sent repeatedly on all the simultaneous connections. This test should indicate the peak performance of the server since the connection setup overheads are eliminated.

Typical performance: To determine the maximum request rate which the server can handle before it reaches saturation when each request is sent on a separate connection. In this test, the client repeatedly sends HTTP requests, opening a new connection for each request. The request rate is increased, starting from 1, till the server saturates. This test does not pipeline requests on a peristent HTTP connection.

6.2 Results

Twenty five runs of the above described tests, for each of the messages, were conducted. The results were measured as the response time per request and the observed service rate i.e., the total number of requests sent divided by the total test time. Figures 3 to 6 depict the mean and the standard deviation of the results obtained.

Figure 3 shows the plot of the response time and the observed request rate with the number of simultaneous persistent HTTP connections for the blank coupon request. As the number of simultaneous connections is increased, the observed service rate remains constant with an approximate value of 20 and the response time increases linearly. This shows that the server can easily support 25 parallel simultaneous connections without getting overloaded.

Figure 4 shows the plot of the response time and the observed service rate with the request rate, for the blank coupon request. For loads of 19 requests/second or less, the server is not overloaded. The observed request rate is equal to the rate at which the requests are sent and the response time is constant. As the load increases above 19 requests/second the server begins to saturate, the response time increases exponentially and the ob-

Operation	Max. number of operations per second
Digital Signatures	86.4
Keyed Hash	3442.0
Web service (using servlets)	120.0
Database (issue)	203.3
Database (verification)	239.8

Table 1: Performance of individual operations.

served request rate levels off.

Figure 5 and 6 shows similar plot of the response time and the observed service rate for the verification request. Here the saturation occurs at the service rate of 10 requests per second. The performance for *verification requests* is lower because this message requires more cryptographic operations (a signature verification and another signature).

6.3 System Scalability

Every year 5-7 billion traditional paper coupons are redeemed in the USA [3]. The performance requirements for a real coupon mint, serving one billion coupons per year, with a peak-to-mean load ratio of 100, is about 3000 verification requests per second. On a PC implementation, the coupon mint was able to support up to 19 blank coupon issue requests per second and and about 10 verification requests per second.

To understand the bottleneck in the system, we divided the system code into the following four logical components: code for cryptographic operations, code for database operations, code for web service related operations, and the prototype electronic coupons code. We wrote small benchmark programs to individually measure the performnace of each of the standard components on the test system. The results are summarized in Table 1.

Carrying out a digital signature using cryptlib with a key size of 1024 bits takes an average of 11.58 ms per signature, whereas a keyed hash function (HMAC-SHA1) on a data of same size using a key of same size takes 290.5 μsec per hash. In the prototype implementation, digital signatures are used to generate unforgeable blank coupons. Therefore, each blank coupon request requires one digital signature and each verification request requires two digital signatures. If keyed hash function is used (as explained in Section 3) instead of digital signatures, this can be reduced to one keyed hash operation for blank coupon issue and one keyed hash and one digital signature for verification request.

We wrote a simple servlet that outputs “Hello World” when invoked. Using a method similar to that used to measure the performance of the prototype system, we measured the performace of this servlet. With this servlet, the test system was able to serve up to 120 requests per second.

To measure the performance of the database system, we wrote sample programs that perform database operations equivalent to those performed during blank coupon issue and blank coupon verification. Table 1 indicates that the the “issue program” is able to perform 203.3 operations per second and the “verification program” is able to perform 239.8 operations per second.

Digital signature is the primary bottleneck that limits the system performance to 86.4 request per second even if the other overheads are optimized to take nearly zero time. One method to speedup this architecture is to use multiple processors to serve the requests arriving in parallel. There are two main issues while parallelizing any program: efficient distribution of the load among different parallel machines, and taking care of data dependencies between different machines.

Fortunately for our architecture the data dependencies are very minimal and therefore it seems amenable to a large-scale distributed implementation. A single coupon mint may deploy several independent subsystems for generating and verifying blank coupons. Each subsystem has its own local database, uses its own key for generating the unforgeable blank coupons, but uses a common coupon mint key to sign the certificate of use. Now, each of these subsystem can function independently, provided the verification requests are always sent to the subsystem that generated the blank coupon corresponding to the request. Thus, while generating a blank coupon, each of these subsystems puts a verification URL in the blank coupon, so that the retailer sends the coupon to the subsystem which generated it. All the blank coupons of the same class (for early bird coupons) are also generated by the same subsystem. Thus, each subsystem functions independently without sharing its database with other subsystems. The load may be partitioned statically across these subsystems based on the identity of the manufacturer and its specific promotion.

Therefore, it seems possible to incrementally scale the coupon mint system by adding independent subsystems as the requirement grows. According to a highly conservative estimate, a production system handling a significant fraction of coupons redeemed in the USA (if all of them were to be converted into electronic coupons) will require a farm of 200 to 400 PC-like systems to deliver adequate performance. This number is expected to go down significantly after the removal of prototyping inefficiencies and with constant advances in processor technologies.

7 Conclusions

In this paper we presented an architecture for secure generation and verification of a variety of electronic coupons. The architecture supports manufacturer and store coupons, targeted and untargeted coupons, limited and unlimited distribution coupons, and variable value coupons of different kinds (early bird, aging, growing, random value coupons).

Separation of coupon issue and distribution from online coupon verification is an important part of this architecture. Thus, individual manufacturers can issue and distribute electronic coupons without requiring an online coupon verification infrastructure of their own. The coupon verification is done using the services of an independent third party coupon mint provider. In this way, different manufacturers can implement their own promotion policies through a variety of available couponing mechanisms.

We have prototyped the electronic coupon system in Java. We briefly described the details of our prototype implementation. We also discussed the design of the retailer component which needs to be integrated with the order processing modules of the online stores.

We reported some preliminary performance measurements on the prototype system. While the performance of the prototype system is limited, we suggested some ways to scale the system for significantly higher performance. We therefore believe that, with appropriate hardware infrastructure it is possible to develop a system which can give a performance required in a production environment.

8 Acknowledgments

Special mention goes to Alok Aggarwal who inspired the authors to carry out this work. We also thank the anonymous USENIX refrees for their meticulous and insightful comments on the paper.

References

[1] P. Kotler, *Marketing Management - Analysis, Planning, Implementation and Control*, Prentice Hall, 1991.

[2] M. Kumar, A. Rangachari, A. Jhingran, and R. Mohan, "Sales promotion on the Internet," in *3rd USENIX Workshop on Electronic Commerce*, (Boston, MA, USA), pp. 167–176, August-September 1998.

[3] Joint Industry Coupon Committee, *Coupons: A Complete Guide*. Grocery Manufacturers of America, 1998, URL: www.gmabrands.com.

[4] S. Brands, "Untraceable off-line cash in wallets with observers," *Advances in Cryptology CRYPTO'93*, pp. 302–318, 1993.

[5] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," *Advances in Cryptology CRYPTO '88*, pp. 319–327, 1988.

[6] N. Ferguson, "Single term off-line coins," *Advances in Cryptology - EUROCRYPT '93*, pp. 318–328, 1993.

[7] M. K. Reiter, V. Anupam, and A. Mayer, "Detecting hit shaving in click-through payment schemes," in *3rd USENIX Workshop on Electronic Commerce*, pp. 155–166, August 1998.

[8] V. Anupam, A. Mayer, K. Nissim, B. Pinkas, and M. K. Reiter, "On the security of pay-per-click and other web advertising schemes," in *8th International World Wide Web Conference*, May 1999.

[9] NIST, "The digital signature standard," *Communications of the ACM*, vol. 35, July 1997.

[10] R. L. Rivest, A. Shamir, and L. A. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[11] O. Kommerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," in *USENIX Workshop on Smartcard Technology*, (Chicago, Illinois, USA), May 1999.

[12] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, "Keyed hash functions," in *Cryptography: Policy and Algorithms Conference*, pp. 201–214, Springer-Verlag, LNCS 1029, July 1995.

[13] C. Adams and S. Farrell, "Internet X.509 public key infrastructure certificate management protocols," RFC 2510, March 1999.

[14] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, "Extensible markup language (XML)," World Wide Web Consortium Recommendation REC- xml-19980210.

[15] T. Dierks and C. Allen, "The TLS protocol version 1.0," RFC 2246, January 1999.

[16] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1," RFC 2616, June 1999.

[17] D. Connolly and L. Masinter, "The 'text/html' media type," RFC 2854, June 2000.

[18] D. Mosberger and T. Jin, "httperf: A tool for measuring web server performance," in *WISP*, (Madison, WI), pp. 59–67, June 1998.