Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2016, Article ID 2349148, 10 pages
http://dx.doi.org/10.1155/2016/2349148

Research Article

An Architecture for Anonymous Mobile Coupons in a Large Network

Alberto Bartoli and Eric Medvet

DIA, University of Trieste, Trieste, Italy

Correspondence should be addressed to Alberto Bartoli; bartoli.alberto@univ.trieste.it

Received 5 September 2016; Revised 28 October 2016; Accepted 15 November 2016

Academic Editor: Gianluigi Ferrari

A mobile coupon (m-coupon) can be presented with a smartphone for obtaining a financial discount when purchasing a product or a service. M-coupons are a powerful marketing tool that has enjoyed a huge growth and diffusion, involving tens of millions of people each year. We propose an architecture which may enable significant improvements over current m-coupon technology, in terms of acceptance of potential customers and of marketing actions that become feasible: the customer does not need to install any dedicated app; an m-coupon is not bound to any specific device or customer; an m-coupon may be redeemed at any store in a set of potentially many thousands of stores, without any prior arrangement between customer and store. We are not aware of any proposal with these properties.

1. Introduction

A mobile coupon (m-coupon) is a piece of data constructed and delivered electronically that can be used for a financial discount when purchasing a product or a service [1]. M-coupons are used for a variety of marketing purposes, usually in sales promotion as well as for attracting customers to services and entertainment [2]. M-coupons have enjoyed a huge growth and diffusion: it has been estimated that more than 97 million people redeemed a mobile coupon in the US in 2015, corresponding to an 18% growth over the previous year [3]. Tens of millions of people redeemed m-coupons by presenting them on the screen on their smartphones or tablets.

From a technological point of view, m-coupon implementations must address several security-related issues. When a customer presents an m-coupon to a merchant for redemption, the merchant must be able to ascertain that the m-coupon (a) is authentic, (b) has not been modified in any part, (c) has not been redeemed already, and (d) has not expired (m-coupons usually have an expiration date). Ensuring these properties is essential to the merchant and, more broadly, to all the actors involved in the marketing action [4].

All the m-coupon implementations that we are aware of are based on one or more of the following constraints: a given m-coupon may be redeemed only by a specific device or a specific customer; a given m-coupon may be redeemed only at a specific store, or at any store in a predefined small set of stores. These requirements are usually fulfilled by forcing customers to handle m-coupons through dedicated software to be installed on their smartphones or tablets [4–6]. Furthermore, customers must often exhibit a validation code or password to the store when redeeming the m-coupon [7, 8].

A key reason for the ubiquitous presence of those constraints is because they allow ensuring the necessary security properties efficiently. For example, an m-coupon inextricably coupled to a certain device cannot be redeemed at multiple locations simultaneously. Similarly, redemption of an m-coupon linked to a small set of stores may be made known at all those stores very quickly.

In this work, we propose the design of an architecture for m-coupons that is not affected by the above constraints. In our proposal, an m-coupon may be redeemed by any device of any customer and an m-coupon may be redeemed at any store involved in the promotion, without any prior arrangement.

The set of such stores may be geographically dispersed and very large, that is, even in the order of many thousands. Most importantly, customers are not required to install any dedicated app on their devices: m-coupons are merely images that can be accessed with any software freely and that may be exhibited to merchants in any form, typically on the screen of a smartphone and also in print. We are not aware of any other proposal with these features.

The resulting scenario has several highly desirable properties. First and foremost, the fact that customers do not need to install any app makes the customer experience easier and removes a crucial constraint that is associated with significant security and privacy-related risks. Indeed, those factors play a crucial role in the willingness of potential customers to participate in marketing actions based on m-coupons [9–11]. Furthermore, the ability to access m-coupons from multiple personal devices freely has nowadays become a practical necessity; for example, one could obtain an m-coupon by email while browsing with a tablet at home and then redeem the m-coupon while travelling with only the smartphone available. The ability to move m-coupons across devices of different owners has become essential as well: families typically have tens of devices, and parents must be able to transfer their m-coupons to kids easily. Removing the need of checking the identity of customers, or their knowledge of a certain code, also enables much faster and frictionless processing of m-coupon redemption within stores. The ability to redeem an m-coupon anywhere without any prior arrangement with the store also contributes to making the customer experience easier: for example, an m-coupon for a cinema chain may be constructed and delivered without forcing the customer to declare in advance at which specific cinema he/she is going to redeem the m-coupon. The simplicity of the overall scenario may encourage potential customers to use m-coupons even when their financial value is small, which may enable a range of marketing actions which do not fit current m-coupon technology well [12–14].

The difficulty in supporting anonymous m-coupons on a large network of stores may be realized easily, as it suffices to consider this key security threat:

(1) A customer obtains a valid m-coupon.

(2) The customer constructs many identical copies of the m-coupon and sends each copy to a set of colluding customers.

(3) All colluding customers present themselves at different stores, at exactly the same time; each store will thus see a customer that exhibits a valid m-coupon.

Note that step (2) may be executed in a matter of seconds and may potentially involve thousands of geographically dispersed colluding customers.

1.1. Related Work. Several m-coupon solutions have been proposed in different forms: patented frameworks, commercial platforms, or research models. To the best of our knowledge, none of them exhibits the properties stated in the previous section. The architecture outlined in [8] requires the customer to have a dedicated application installed on his/her smartphone, while the one in [15] binds each m-coupon to a specified customer device. The commercial platform described in [7] may be deployed in a variety of ways. Validation of an m-coupon may be done either on a dedicated application installed on the customer device or on a different dedicated application installed on a device in control of the store. In the latter case, the customer has to provide a password to be manually inserted in the device. The password must be made known to the store in advance and is bound to a single store/m-coupon pair. Fulfilling this requirement in a large network of geographically distributed stores is clearly difficult. A centralized service has to be contacted in order to prevent single spending. Depending on the specific deployment options, the centralized service is also the only entity able to check authenticity, integrity, and temporal validity. Similar remarks apply to platform [5]. Platform [6] is designed for retailers composed of a small number of shops and stores m-coupons in a dedicated application on the customer device.

Models and protocols for generating and distributing digital coupons with strong security properties are proposed in [4]. The analysis is not focused on the technological aspects and the proposed implementation assumes that the customer is equipped with a dedicated application.

A similar remark applies to more recent proposals in [16, 17], which assume that the customer is equipped with a dedicated application able to implement a certain communication protocol.

Finally, we mention that the ability of our architecture to validate authenticity of an m-coupon locally (i.e., without contacting a centralized store) even when the m-coupon is not bound to the location of the validating entity may be useful in other application domains. For example, it has been recently shown that exclusive airline lounges in several airports allow access by presenting a simple QR code that can be faked easily [18].

2. Our Approach

2.1. Problem Definition. An m-*coupon* is a piece of data that is constructed by an *issuer* and may be redeemed by a *customer* at a *store* in exchange for an item or a service. The customer does not need to install any dedicated app and exhibits an m-coupon either in print or on the screen of a smartphone.

An m-coupon has a predefined expiration date and is not bound to any specific customer or store. We target a design region consisting of potentially thousands of stores, where thousands of m-coupons may be redeemed at each store each day. Transmission of m-coupons from issuer to customers is orthogonal to this work, as well as all the business and operational agreements between issuer and stores.

Our proposed approach ensures the following security properties for m-coupons: (a) *authenticity* (an m-coupon can only be constructed by the issuer), (b) *integrity* (any modification to an m-coupon is detected upon redeeming), (c) *single spendability* (an m-coupon cannot be redeemed more than once), and (d) *temporal validity* (an m-coupon cannot be redeemed after its predefined expiration date).

In this section, we assume that attacks on these security properties may only come from customers: we assume that stores and technical infrastructure are trusted and we shall relax these assumptions in a later section. Concerning attacks on the single-spendability property, in particular, we consider the worst possible threat model: many redeeming attempts of the same m-coupon may occur simultaneously at many different, geographically dispersed stores. Of course, the more pessimistic the threat model, the more potentially complex and costly the solution.

We emphasize our assumption that customers do not need to install any dedicated app on their smartphones or tablets. While assuming that customers exhibit m-coupons through a dedicated trusted app would simplify the implementation, we believe that considering customer-provided data as untrusted leads to a stronger solution:

(1) The app might be reverse-engineered and/or its security properties might be circumvented.

(2) It would be very difficult for a clerk to make sure that the app shown by a customer is indeed the legitimate app.

Furthermore, forcing customers to install an app on their smartphones or tablets would prevent the redemption of printed coupons and may hinder wide acceptance of the solution; such a requirement is intrusive and constitutes a strong privacy risk.

2.2. Usage Scenario. In our approach, an m-coupon consists of two components: a QR code and a description of the item associated with the m-coupon. The clerk at the store is equipped with a dedicated *validation app* running on a smartphone. The validation app may be distributed and updated through the standard app store mechanisms (i.e., Google Play, iTunes, and Windows Store). The smartphone needs an Internet connection.

The clerk scans the QR code in the m-coupon exhibited by the customer and the validation app quickly shows one of the following outcomes:

(1) An indication that the m-coupon may be redeemed, along with a short textual description of the associated item. This description may include a universal product identifier (https://en.wikipedia.org/wiki/Global_Trade_Item_Number) to simplify integration with store and register working.

(2) An indication that the m-coupon cannot be redeemed, along with a textual description of the reason, which may be one of the following:

   (a) The m-coupon is not authentic or has been altered or has expired.

   (b) The m-coupon has been redeemed already.

   (c) The m-coupon cannot be redeemed in this moment for technical reasons.

The full operation will take a few seconds, similar to a credit/debit card payment (see also the next section). Note that stores do not need to ask for any proof of identity of customers.

Clerks must be instructed to do the following:

(1) Never scan QR codes with a customer-provided device. They must use only a store-provided smartphone (or their personal smartphone, if allowed to do so by the store owner).

(2) Trust only the textual description coming from the app after scanning the QR code. The item description presented by the customer in an m-coupon is not to be trusted.

The app verifies authenticity, integrity, and temporal validity of the coupon *locally*. The app verifies that the coupon has not been *redeemed already* by contacting via HTTPS a *centralized service*.

2.3. Implementation. Key feature of our approach is that the QR code embeds a *cryptographic signature* of the issuer. It follows that m-coupons cannot be forged or modified. The validation app ensures authenticity, integrity, and temporal validity *locally*, by simply verifying the validity of the signature. Contemporary technology makes this approach practically feasible, even with low-end smartphones (see Discussion). The validation app cannot ensure the single-spendability property locally: it contacts a *centralized service* which serializes m-coupon redemptions for preventing coordinated attacks aimed at multiple redemptions of the same m-coupon at different stores.

2.3.1. QR Code and App. The QR code is a piece of data cryptographically signed by the issuer. In detail, the QR code encodes a sequence of data that we denote as <D,S>; D consists of the following:

(i) D-code: a pseudorandom code that uniquely identifies each coupon and is sufficiently long to not be guessed (e.g., 20 characters)

(ii) D-text: a short textual description of the item associated with the coupon, optionally including a universal product code

(iii) D-expire: expiration date of the coupon

while S is the corresponding signature of D (full details in the next section).

The validation app executes the following operations:

(1) Scan the QR code, extracting <D,S>.

(2) Verify authenticity, integrity, and temporal validity of the coupon. This operation is done locally.

   (i) Authenticity and integrity are verified by simply verifying that S is indeed a valid signature for D. To this end, the app must have a certificate with the public key of the signer. This certificate may be distributed with the app.

   (ii) Temporal validity may be verified by simply comparing D-expire to the current date.

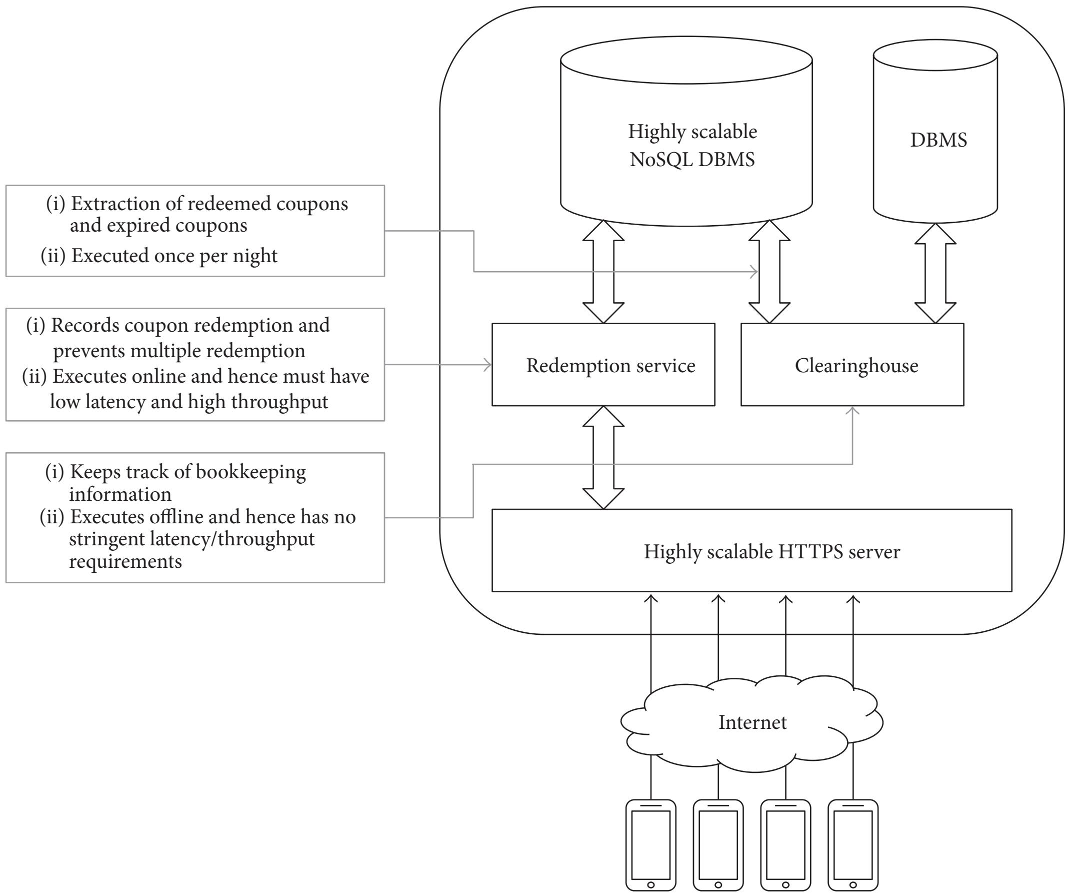


Figure 1: Centralized service architecture.

(3) In case of failed verification, terminate with outcome (2)-(a) in the previous section; otherwise, do the following.

(4) Send D-code to a central location and receive either

   (i) "ok" corresponding to outcome (1) in the previous section,

   (ii) "not-ok" corresponding to outcome (2)-(b).

All these operations are typically executed in a few seconds (in particular, the cryptographic steps involved in the digital signature verification and HTTPS interaction with a centralized server may be expected to take much less than one second based on the data in [19, 20]). These operations can be easily integrated with the usual store and register working. Communication exceptions are presented by the app as outcome (2)-(c) in the previous section.

Note that the QR code is not used as a URL-encoding mechanism: the validation app always connects to the same service whose address is hardwired within the app itself. It follows that phishing or pharming attacks in which a customer drives the validation app on a fake web site under the control of the customer are impossible [21].

The app may support a working mode in which the centralized service may only be queried without redeeming the coupon. This working mode will be used only in special cases, that is, for ascertaining the outcome of a transaction which experienced a communication failure, or for answering occasional questions by customers that are unsure whether a coupon is still valid or has been redeemed already.

2.3.2. Centralized Service. The centralized service is composed of two main components (Figure 1): a *redemption service*, to be accessed by each retailer in order to record redemption of each m-coupon and prevent multiple redemptions of the same m-coupon, and a *clearinghouse*, which associates each redeemed coupon with the corresponding retailer service. We emphasize that the design has been carefully structured to accommodate the different latency and throughput requirements of the two components:

(i) The redemption service executes only one single-row atomic transaction for each m-coupon redemption. This allows using *highly scalable* NoSQL databases. These complex multirow/multitable transactions are not required at coupon redemption time.

(ii) The clearinghouse typically requires multirow/multitable transactions but these transactions are not executed synchronously as part of coupon redemption by the customer. Thus, this component may rely on more traditional database technologies without any critical requirement in terms of latency and throughput.

```
CheckAndUpdate (D-code):
RetVal ≔ not-ok;
Begin Atomic Transaction
     row ≔ row of CouponCodeStatusTable such that row.CouponCode = D-code;
     if (row <> null AND row.RedeemedStatus == false)
     then
          row.RedeemedStatus ≔ true;
          row.RedeemedBy ≔ storeId;
          row.RedeemedWhen ≔ now;
          RetVal ≔ ok
End Atomic Transaction
return RetVal
```

Algorithm 1: Pseudocode for request processing at the centralized service. The `storeId` value is an attribute of the authenticated HTTPS session.

The main data structure at the redemption service is the following:

(i) CouponCodeStatusTable: a table with one element for each coupon that has not been redeemed and is not yet expired. Each element is a tuple:

(a) CouponCode: the D-code encoded by the QR code

(b) RedeemedStatus: a Boolean

(c) RedeemedBy: unique identifier of the app instance which redeemed the coupon

(d) RedeemedWhen: date of redemption

Redeemed and expired m-coupons are removed periodically at times of light load (e.g., each night). M-coupons just created are inserted into the table by the same mechanism.

The CouponCodeStatusTable is accessed for the processing of requests from the app, that is, during customer transactions at the store. Thus, access to this table must be very efficient. This table is implemented with a NoSQL database supporting atomic read-write update at the level of a *single* row [22]: a row will switch from RedeemedStatus = false to RedeemedStatus = true (along with the corresponding values for RedeemedBy and RedeemedWhen) only if RedeemedStatus is false.

Complex multirow/multitable atomic transactions may be avoided because the QR code is cryptographically signed: authenticity, integrity, and temporal validity are guaranteed by the cryptographic signature on the QR code, and thus transactional access is required only for preventing multiple redemptions of the same m-coupon. Furthermore, the database engine will always be able to execute at maximal throughput because concurrent access to the same row is absent—multiple access to the same row may occur only when the same m-coupon is scanned at different places at the same time.

The clearinghouse component will maintain auxiliary data structures for keeping track of stores, emitted m-coupons, credentials, and so on. These data structures are not accessed during customer transactions, and thus efficiency of access is not critical and a broad range of technologies may be used. We omit those data for ease of presentation.

A back-end process will run each night for moving coupon information from the redemption service to the clearinghouse. Value of the RedeemedBy field in the CouponCodeStatusTable will be used, perhaps through further intermediate tables, for performing the necessary join operations.

2.3.3. Communication. Communication between app and centralized service occurs on an HTTPS channel, with only one application-level message round. A request from the app consists of a pair <D-code, operation> where operation may be either `CheckAndUpdate` or `Check`. The corresponding response may be either "ok" or "not-ok" as described above. The processing of each request is described in Algorithm 1.

In terms of the lightweight transactions supported in Cassandra (a highly scalable NoSQL database; see also the next section), the single-row atomic transaction may be expressed as [22]

```
UPDATE CouponCodeStatusTable SET
RedeemedStatus = true,
RedeemedBy = storeId, RedeemedWhen = now
    WHERE CouponCode = D-code
    IF RedeemedStatus = true;
```

The `Check` operation may be executed without any transaction (we omit its description for simplicity). We emphasize that single-row atomic transactions of this form are not a peculiarity of Cassandra as they are supported in other NoSQL databases such as, for example, MongoDB and HBase.

Each app must authenticate itself to the centralized service. Lifecycle management of credentials is to be performed by the issuer. In order to not perform authentication upon each connection, the standard practice of mobile application development may be used, in which credentials are inserted in the app upon first use and then the app is implicitly authenticated indefinitely (or until revocation from either side). The software technology is the "implicit grant" in the OAuth2 framework [23].

3. Discussion

3.1. QR Code. QR codes are routinely used for ticket verification by airlines and public transport companies (e.g., [24]). This fact demonstrates that scanning a QR code in an "unfavorable" environment is indeed feasible and practical.

In those scenarios, though, authenticity, integrity, and single spendability may be guaranteed easily: a QR code may contain just a short code to be checked against a database locally available to the scanning device (although there are many real systems which fail to implement these steps correctly [18]). In the scenario of our interest, this approach is not feasible because m-coupons are not bound to any specific location. Thus, it would be necessary to distribute a description of all newly emitted m-coupons to many thousands of geographically dispersed databases; and, most importantly, each redemption would have to be recorded instantaneously at all those databases to prevent frauds by multiple colluding attackers presenting the same QR code at different geographical locations.

To demonstrate that a QR code may easily store the larger amount of information required in our scenario, consider what follows. The QR code must encode a triplet <D-code, D-text, D-expire> along with an associated signature S, as described previously.

Signature S could be generated, for example, with RSA signature. The length of an RSA signature is the same as the length of the key size used for signing. A 2048-bit key size is the current standard in most environments [25]. A signature generated with such keys will thus require 256 alphanumeric characters.

Field D-expire may be represented with 6 characters while D-code could be represented with 20 bytes. It follows that we need a QR code capable of encoding 256 + 6 + 20 + size of (D-text) = 282 + size of (D-text) characters.

Data capacity of a QR code depends on the "version" of the QR code, that is, on the specific format and coding used. A QR ode with version 14, error correction M can store 528 alphanumeric characters (http://blog.qrstuff.com/2011/12/14/qr-code-error-correction). Error correction M is the strongest of the two levels designed for general marketing use. In this case, there would be 528 − 282 = 246 characters available for D-text. Even assuming that D-text contains a form of universal product code (https://en.wikipedia.org/wiki/Universal_Product_Code), there would be an extra 230 characters left. Information in the D-text field is thus more than enough for enabling a clerk to reliably identify the item associated with the mobile coupon.

Next, we need to demonstrate that those QR codes may indeed be decoded quickly and reliably with simple devices. To this end, consider what follows.

Generation of a QR code with the desired "version" can be done with a variety of tools, including online web apps (e.g., http://www.morovia.com/free-online-barcode-generator/qrcode-maker.php). We generated QR codes of the above kind, with a "module width" of 2 mm. Tests with the camera of a low-end smartphone (Motorola Moto G 2014 (https://en.wikipedia.org/wiki/Moto_G_(2nd_generation))) equipped with an off-the-shelf QR code reader app (QR Code Droid Scanner https://play.google.com/store/apps/details?id=la.droid.qr&hl=en) showed quick and reliable decoding:

(i) From a screen showing the QR code in size approx. 4 cm (i.e., the web site that generates the QR code)

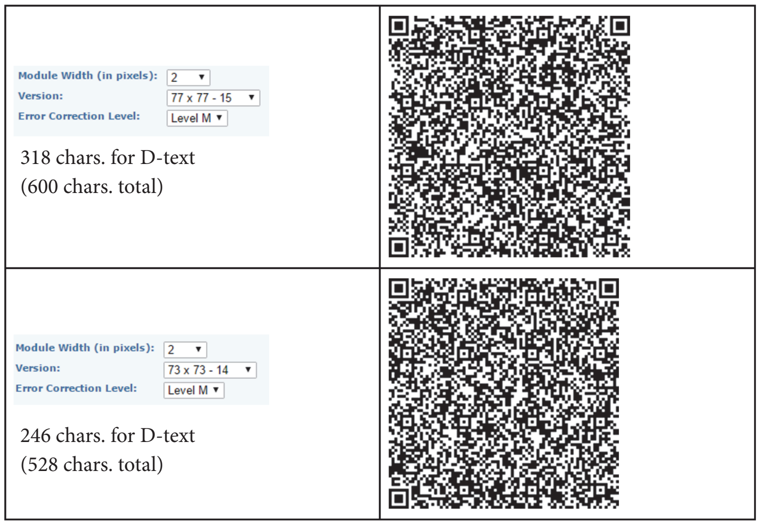
(ii) From a printed sheet generated with a very low-end laser printer, provided the QR code is at least 3 cm

Tests with QR code version 15, which allows storing an additional 80 characters (600 characters), gave identical results.

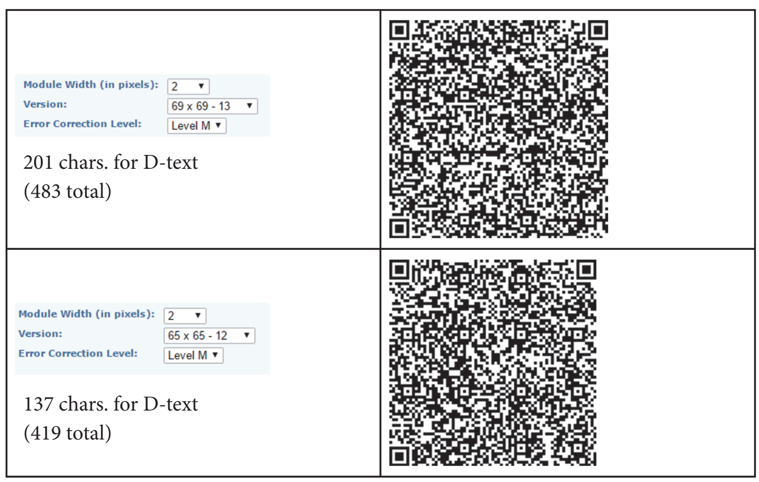
Figure 2 contains four examples differing in the QR code version used, which results in different amounts of characters available for D-text (indicated in the table itself). All the examples use 6 characters for D-expire, 20 characters for D-code, and 256 characters for the signature. Of course, one might choose to distribute the characters available for the D-fields differently. We invite the reader to scan the examples, on screen and in print.

3.2. Centralized Service. The centralized service implementation certainly requires careful engineering to meet the desired requirements in terms of scale, latency, and availability. However, the service is practically feasible based on the following considerations. Assuming a number of stores in the order of tens of thousands (e.g., 100,000) and hundreds of redeemed m-coupons per store per day (e.g., 500), the service has to be able to sustain throughput in the order of $5 \cdot 10^7$ redemptions per day. Provisioning the system requires an estimation of the maximum number of stores connected per second. Assuming that all m-coupon redemptions are fully concentrated in only 5 hours, one would need to sustain 100 coupons per hour per store, that is, less than 2 coupons per minute per store. Assuming that all stores inject such a maximum load for the 5 hours, the centralized service would need to be able to sustain 200,000 interactions per minute, that is, less than 3,400 per second. Below, we argue that such a performance is easily achieved with off-the-shelf technology.

The key functionality required by the database engine is single-row atomic transactions. A recent benchmarking of several NoSQL databases has included this functionality ("read-modify-write workload, YCSB type F") [26]. The Apache Cassandra database has been tested with up to 32 nodes and the throughput has grown linearly reaching 200,000 operations per second. The workload consisted of 50% read and 50% read-modify-write ratio: since coupon redemption corresponds to 1 read-modify-write operation, we may estimate that 100,000 redemptions per second may be sustained. It is fair to claim that this figure is more than enough for sustaining the expected load. Also, recall that row-level conflict will be absent most of the time (except in case of fraudulent attempts to redeem the same coupon at the same time); hence, the optimistic concurrency control mechanism used in Cassandra will be able to drive the engine at maximal throughput (no locking overhead, no abort-rollback). Latency ranged from 30 ms (1 node) to 41 ms (32 nodes), also in line with the requirements. There are other reports indicating the ability of Cassandra to scale linearly up to more than 100 nodes and 1,000,000 writes per second



(a)



(b)

Figure 2: (a) Examples of QR codes encoding the required information (version 15 and version 14). (b) Examples of QR codes encoding the required information (version 13 and version 12).

[27, 28], with one single Amazon EC2 instance able to sustain more than 10,000 writes per second.

Concerning the HTTPS engine, a recent benchmarking of the NGINX web (proxy) server with modern cipher suites is summarized as follows [29]: "*NGINX's SSL performance scales with the number of cores available on the host server, until other limits (typically bandwidth) are met, so an 8-core virtual machine could accept traffic from over 1,000 new users per second and still have resources to spare.*" Based on the abovementioned benchmarks, thus, a few NGINX boxes with a load balancer are more than enough.

3.3. Security Properties. Authenticity, integrity, and temporal validity are guaranteed by the cryptographic signature of QR codes: as long as the cryptographic procedures are implemented and deployed correctly (i.e., the signing key is indeed private and signatures are indeed verified with the corresponding public key), forging a QR code that will be considered as valid by the scanning app is impossible. Single spendability is guaranteed by the fact that all redemptions are serialized by the centralized service.

We have assumed so far that attacks may come only from customers which attempt to forge an m-coupon or modify an existing m-coupon or to redeem the same m-coupon multiple times. Below, we will consider other forms of attack. We do not consider the case in which an attacker (either a customer or a clerk) steals an existing valid m-coupon: this threat cannot be avoided because m-coupons are not bound to any owner.

3.3.1. Network Attacks. HTTPS is the standard de facto for any Internet-based secure interaction. Correct HTTPS deployment and usage ensure authentication of the service, as well as integrity and secrecy of the communication between app and centralized service. An attacker could attempt to hijack the communication channel between app and centralized service, that is, by attempting to impersonate the service and reply with "ok" either to every request or only to selected requests. The best practices in HTTPS deployment and app programming constitute very effective defenses in this respect. In particular, the app must be coded so as to connect only to the URL of the centralized service, that is, without allowing usage of the HTTP protocol and without allowing connecting to different URLs. Furthermore, the app must implement certificate validation correctly, in particular, without accepting self-signed server certificates.

3.3.2. Misuse of the Legitimate App. The validation app is distributed through the public app stores, in order to ensure authenticity and integrity of the code, as well as to enable smooth and seamless distribution of app upgrades and fixes.

An attacker installing the app would not be able to interact with the service, because he would not have the necessary authentication credentials. An attacker using the credentials of a store could only redeem an existing valid m-coupon (a stolen one). In that case, the redemption would provoke a gain for the store and the customer could not obtain the item associated with the coupon. Thus, there is no incentive for customers in pursuing these attacks.

3.3.3. Usage of a Fraudulent App. An attacker could implement a fraudulent app able to communicate with the service. This attack requires reverse-engineering of the application protocol used by the legitimate app and valid authentication credentials. The fraudulent app may only

(1) send coupon codes generated at random;

(2) send coupon codes extracted by a signed QR code, without performing any check on the signature.

Assuming coupon codes are implemented correctly (i.e., they are truly random and sufficiently long to not be guessed), point (1) is a nonissue. Of course, simple checks should be implemented at the centralized service for detecting an excessive number of failed attempts from a session and reacting appropriately.

Concerning point (2), the fraudulent app may send the coupon code extracted from

(A) a valid, possibly stolen m-coupon;

(B) an m-coupon that has expired and has not been redeemed.

Point (A) is a nonissue (in this case, there would be no point in implementing a fraudulent app; the legitimate app could be used). Point (B) would not be detected by the service while interacting with the fraudulent app, but it would be detected later by the clearinghouse component (based on the AllCouponsTable outlined in the Implementation). Consequently, the clearinghouse component would not include the coupon in the store account. Note that attacks of this sort could be meaningful only to clerks or store owners: customers would have no reasons for implementing a fraudulent app, as they could not obtain the item associated with the coupon.

3.4. Signing of QR Codes. Since the coupon code is sent to the centralized service at each redemption, the additional coupon information (item description and temporal validity) could be sent back by the service. One may thus wonder why a QR code contains that information, which also introduces the need for a digital signature: a QR code containing only a coupon code would suffice. In that case, checking for authenticity, integrity, and temporal validity would be all demanded to the centralized service, rather than being delegated to the app.

There are several reasons why the QR code is cryptographically signed:

(i) It prevents transmission of forged, altered, or expired m-coupons to the service; this property eliminates a number of possible attack vectors based on QR codes, for example, command injections or attempts to exploit buffer overflows at the server side [21].

(ii) It minimizes the risk of introducing security vulnerabilities due to any possible change in the m-coupon format (e.g., addition of further fields during solution development or maintenance).

(iii) It minimizes the risk of introducing security vulnerabilities in the communication protocol (which is extremely simple: one single piece of information, one single round, or one single check).

(iv) It increases the perceived security and robustness of the solution by stores: even if they cannot enter into the technical details, the mere fact that a mobile coupon is "cryptographically signed" is certainly very important in this respect.

Furthermore, usage of cryptographically signed QR codes minimizes the amount of data to be transmitted over the HTTPS channel. Using the same data size as in the previous analysis (D-code 20 characters, D-expire 6 characters, and D-text 230 characters), one has the following:

(i) Proposed solution: a request is a D-code and an operation description; a response is an operation outcome; there are approximately 20 characters in total.

(ii) Solution without cryptographic signature: a request would be identical, while a response would contain also D-expire and D-text; there are approximately 230 characters in total.

Saving 200 characters for each request may be important because all service interactions occur over HTTPS; that is, they are encrypted. Throughput in terms of bytes/sec, rather than in terms of clients serviced, may often be an issue. Although the actual saving will be amortized by the fixed cost of HTTP headers, the difference between 20 and 230 characters is not negligible. The amount of data involved in disk transfers is also minimized in the same way.

3.5. Single Spendability. In order to ensure single spendability, the proposed approach is based on a *centralized-synchronous* architecture: a centralized database in which each redemption is recorded immediately. Indeed, this is the approach taken in most existing systems with the crucial observation that, in those systems, a given m-coupon may be redeemed only at a specific store, or at any store in a predefined small set of stores [4–8]. Alternative architectures may be devised which differ in terms of *when* and *where* redemptions are stored. We elaborate on those architectures below.

In a *centralized-asynchronous* architecture, redemption would be recorded on the centralized database some time after completion. This architecture would introduce a temporal window of vulnerability in which the same authentic and valid coupon may be redeemed at multiple different stores. In other words, this design would not strive to prevent frauds: it would require crossing fingers in the hope that the frauds that are technically permitted do not occur too often. We reject this design because of the following:

(1) The issuer would have to take the full risk involved in multiple coupon redemptions; otherwise, stores would not accept to be involved in m-coupons.

(2) This risk does not seem to provide any technical advantage: shrinking the vulnerability window down to a few minutes would not simplify the centralized service implementation significantly.

A variety of *distributed* architectures for the service implementation are possible. For the purpose of this discussion, all such implementations may be captured by assuming that the service consists of a number of *service points*, abstracting away the details related to the information stored at each service point and to the coordination between them.

In a *distributed-asynchronous* architecture, redemption would be recorded at some service point and the actual customer transaction may be complete before all necessary service points are updated. Such an architecture is similar to the centralized-asynchronous one, except that the systematic window of vulnerability may be much larger, due to the need of updating multiple service points. A *distributed-synchronous* architecture would be similar, with the added inconvenience that customer transactions would take longer to be complete because all service points have to be updated before completion.

A design in which each store has a locally available list of m-coupons already redeemed at other stores would fall in the distributed-asynchronous category. In this case, stores would have to be willing to work with information that is updated in periods of tens of minutes or of a few hours: the corresponding window of vulnerability would be far too long. Furthermore, it would be easy for stores to realize this intrinsic weakness, which would severely and negatively affect acceptance of the system.

4. Concluding Remarks

We proposed an architecture for an m-coupon system and analyzed its practical feasibility in detail. M-coupons consist of a cryptographically signed QR code. The architecture provides strong security guarantees while at the same time providing high flexibility, high scalability, and ease of implementation and deployment.

Consumers are not required to install any software on their devices and are free to handle m-coupons as any other electronic image, including the ability to move them across their devices as well as across devices of their relatives. An m-coupon may be redeemed by any device of any customer; and an m-coupon may be redeemed at any store, without any prior arrangement between the consumer and the store. The architecture may accommodate tens of thousands of geographically dispersed stores easily. Stores may validate m-coupons with a low-end smartphone equipped with a validation app, which guarantees authenticity, integrity, and temporal validity locally and single spendability through a one-way HTTPS interaction with a centralized service. Single spendability is guaranteed without any temporal window within which m-coupons could be redeemed by colluding attackers at different stores.

The resulting scenario may enable significant improvements over current m-coupon technology, in terms of acceptance of potential customers and of marketing actions that become feasible.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

References

[1] Introduction to Mobile Coupons (version 97), Mobile Marketing Association, 2007, https://www.mmaglobal.com/files/mobilecoupons.pdf.

[2] P. Reichhart, C. Pescher, and M. Spann, "A comparison of the effectiveness of e-mail coupons and mobile text message coupons for digital products," *Electronic Markets*, vol. 23, no. 3, pp. 217–225, 2013.

[3] Marketers Boost Efforts to Reach Coupon Clippers Via Mobile—Emarketer, In: Emarketer.Com, http://www.emarketer.com/Article/Marketers-Boost-Efforts-Reach-Coupon-Clippers-via-Mobile/1012488.

[4] C. Blundo, S. Cimato, and A. De Bonis, "Secure e-coupons," *Electronic Commerce Research*, vol. 5, no. 1, pp. 117–139, 2005.

[5] "4Easy Validation Methods," In: qr2coupon, http://www.qr2-coupon.net/en/how_to/validate_a_coupon.

[6] Mobile Coupon Software for Retailers. In: Codebroker [Internet], http://codebroker.com/mobile-coupon-software.

[7] "Mobile Coupon Validation-How To Validate Mobile Coupons?" In: Coupontools, http://www.coupontools.com/en/how-to-validate-mobile-coupons.

[8] Mobile Coupon Redemption System, US Patent. 20110302017, 2011.

[9] N. M.-J. Achadinha, L. Jama, and P. Nel, "The drivers of consumers' intention to redeem a push mobile coupon," *Behaviour and Information Technology*, vol. 33, no. 12, pp. 1306–1316, 2014.

[10] S. Lee, J. Hwang, and M. Y. Hyun, "Mobile services as a marketing tool to enhance restaurant revenue: an exploratory study," *Journal of Hospitality Marketing & Management*, vol. 19, no. 5, pp. 464–479, 2010.

[11] N. Limpf and H. A. M. Voorveld, "Mobile location-based advertising: how information privacy concerns influence consumers' attitude and acceptance," *Journal of Interactive Advertising*, vol. 15, no. 2, pp. 111–123, 2015.

[12] R. A. Clark, J. J. Zboja, and R. E. Goldsmith, "Antecedents of coupon proneness: a key mediator of coupon redemption," *Journal of Promotion Management*, vol. 19, no. 2, pp. 188–210, 2013.

[13] A. Dickinger and M. Kleijnen, "Coupons going wireless: determinants of consumer intentions to redeem mobile coupons," *Journal of Interactive Marketing*, vol. 22, no. 3, pp. 23–39, 2008.

[14] F. Liébana-Cabanillas, I. Ramos de Luna, and F. J. Montoro-Ríos, "User behaviour in QR mobile payment system: the QR Payment Acceptance Model," *Technology Analysis and Strategic Management*, vol. 27, no. 9, pp. 1031–1049, 2015.

[15] Mobile Coupon Redemption System, US Patent. US20070-156517, 2007.

[16] H.-C. Hsiang, "A secure and efficient authentication scheme for M-coupon systems," in *Proceedings of the 8th International Conference on Future Generation Communication and Networking (FGCN '14)*, pp. 17–20, IEEE, Haikou, China, December 2014.

[17] C.-C. Chang and C.-Y. Sun, "A secure and efficient authentication scheme for e-coupon systems," *Wireless Personal Communications*, vol. 77, no. 4, pp. 2981–2996, 2014.

[18] A. Greenberg and L. Newman, Fake Boarding Pass App Gets Hacker Into Fancy Airline Lounges. In: WIRED [Internet], https://www.wired.com/2016/08/fake-boarding-pass-app-gets-hacker-fancy-airline-lounges/.

[19] Y. Shin, M. Gupta, and S. Myers, "A study of the performance of SSL on PDAs," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM '09)*, April 2009.

[20] H. Tschofenig and M. Pegourie-Gonnard, "Performance of state-of-the-art cryptography on ARM-based microprocessors," in *Proceedings of the NIST Lightweight Cryptography Workshop 2015*, Gaithersburg, Md, USA, 2015 http://csrc.nist.gov/groups/ST/lwc-workshop2015/presentations/session7-vincent.pdf.

[21] P. Kieseberg, M. Leithner, M. Mulazzani et al., "QR code security," in *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia (MoMM '10)*, pp. 430–435, ACM, Paris, France, November 2010.

[22] J. Ellis, Lightweight transactions in Cassandra 2.0 [Internet], Datastax, http://www.datastax.com/dev/blog/lightweight-transactions-in-cassandra-2-0.

[23] D. E. Hammer-Lahav and D. Hardt, "The oauth 2.0 authorization protocol," IETF Internet Draft, 2011.

[24] L. Finžgar and M. Trebar, "Use of NFC and QR code identification in an electronic ticket system for public transport," in *Proceedings of the 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM '11)*, pp. 1–6, IEEE, Split, Croatia, September 2011.

[25] P. Ducklin, Anatomy Of A Change—Google Announces It Will Double Its SSL Key Sizes, Naked Security, https://nakedsecurity.sophos.com/2013/05/27/anatomy-of-a-change-google-announces-it-will-double-its-ssl-key-sizes/.

[26] Benchmarking Top NoSQL Databases [Internet], "End Point," May 2015, http://www.datastax.com/wp-content/themes/datastax-2014-08/files/NoSQL_Benchmarks_EndPoint.pdf.

[27] Cockcroft Adrian And. Benchmarking Cassandra scalability over AWS: over 1 million writes per second [Internet], Netflix, Novmber 2011, http://techblog.netflix.com/2011/11/benchmarking-cassandra-scalability-on.html.

[28] C. Kalantzis, "Revisiting 1 Million Writes per second [Internet]," Netflix, July 2014, http://techblog.netflix.com/2014/07/revisiting-1-million-writes-per-second.html.

[29] NGINX SSL Performance [Internet], NGINX, July 2014, https://www.nginx.com/wp-content/uploads/2014/07/NGINX-SSL-Performance.pdf.