EFFICIENT ENCODING AND TRANSMISSION OF DIGITAL RECEIPTS FOR MOBILE COMMERCE

Matthias Raffelsieper, Alexander Ilic, Thorben Keller, and Elgar Fleisch

ITEM-HSG, University of St. Gallen, Dufourstrasse 40a, 9000 St. Gallen, Switzerland

Email: {Matthias.Raffelsieper,Alexander.Ilic,Thorben.Keller,Elgar.Fleisch}@unisg.ch

Abstract

Making receipts digital creates a new dialog between customers, retailers, and brands, allowing them to discuss products and purchases in real-time. However, the transmission of digital receipts is still a problem, since often Internet connectivity is not available at the point of sale which hampers a real-time interaction. To overcome this problem with current technology, this paper presents a way to efficiently transmit a complete receipt in a QR-code, a certain kind of 2D matrix code (often also called "2D barcode"). Thereby, only a smartphone equipped with a camera is needed and Internet connectivity is not a problem anymore. However, due to size constraints, the encoding of a full receipt needs to be as efficient as possible. We present a custom, domain-specific encoding that was developed exactly for this purpose and show that our prototype implementation performs better than sophisticated, general purpose compression algorithms on this kind of data.

1 Introduction

Smartphones are currently revolutionizing the interaction among users and their surroundings, connecting them via online services over the Internet. Studies, such as for example the German Go Smart 2012 study (Google Inc., Otto Group, TNS Infratest, and Trendbüro 2012), have shown that most of the time a user interacts with such a device is spent on activities other than making phone calls and that due to these devices users are not only online at home, but are ubiquitously connected to the Internet's virtual world and want to stay in touch with their peers. Furthermore, smartphones are equipped with numerous different sensors and input devices, making them the ideal method of interacting between the virtual and physical world. Additionally, they are almost always carried by the users (75 % of all users never leave home without their smartphone, according to (Google Inc., Otto Group, TNS Infratest, and Trendbüro 2012)), making them ubiquitously available. Even though smartphones are usually connected to the Internet via wireless technologies, such as WIFI or mobile data connections, in certain situations none of these are available. This especially holds inside retail stores that are often constructed using metal and thereby often make mobile data connections unavailable. Recently, smartphone apps for managing receipts have received significant attention and are increasingly popular. At the same time, traditional retailers started providing receipts digitally for warranty and customer service purposes. However, the best practice today is focused on digital receipt delivery via email as PDF/html document.

The key issues with the present solution are as follows. First, the receipt data should be encoded in a way that a wide range of smartphone apps can use the data for analytics, tracking, sharing, and recommendations with the user in control of sharing settings. Second, receipt delivery must be reciprocal to the payment to enable comparisons and timeliness. Third, the solution should be pervasively applicable at low-cost for both, retailers and consumers.

To address these issues, we present a novel approach using a solution of efficiently encoding receipts with 2D matrix codes, that is low-cost since it only uses the camera present in virtually every smartphone that is in use today. It displays a receipt encoded as a QR-code (ISO/IEC Std. 18004 2006), a certain instance of these types of codes with an appealing feature set, on a customer display

and/or on a paper receipt. On a technical level, we particularly focus on the aspect of performance and efficiently encoding receipt data in the QR-code. An analysis of over 200,000 real-world retail receipts revealed that present compression and storage algorithms for QR-codes are not sufficient. Therefore, we developed a new, retail domain-specific encoding that leverages frequencies based compression. Even in application scenarios of short receipts, our algorithm proves to be critical since the usability for QR scanning depends on the density of information stored.

The rest of the paper is structured as follows. In Section 2, we discuss existing research into the topic of customer interaction based on product data and research regarding existing barcode encoding and scanning technologies. Afterwards, Section 3 presents our solution to encode receipts as QR-codes. To assess the efficiency of our encoding, we compare it with existing, general purpose compression algorithms in Section 4, where it will be shown that our customized approach works better than these algorithms on the specific type of data contained in receipts. Finally, Section 5 concludes the paper and presents some benefits and use-cases of digital receipts.

2 Related Work

Consumers are increasingly using smart phone applications for discussing products and product characteristics. Such applications have already been created and have sparked great interest (Karpischek & Michahelles 2010, von Reischach, Dubach, Michahelles & Schmidt 2010). They do not only allow users to get recommendations by electronic word-of-mouth (Dellarocas 2003), but it has also been shown that these interactions increase the likeliness of purchases (Senecal & Nantel 2004). However, these applications often require manual data entry and do not include the retailers and brand owners in the dialog.

Research on how to bridge the gap between supply chain data of products beyond the point-of-sale via digital receipts is scarce. The current state-of-the-art in retailing is to deliver electronic receipts through downloads or via email. In most cases, the data is encoded in html or PDF documents and cannot be easily and timely used by apps.

To connect this real-world event with the virtual world, we make the receipt digital and transmit it to the user's smartphone. However, to accomplish this, we need to find a solution that directly transmits it, even though no Internet connectivity can be assumed at the point of sale, as discussed above. A solution that seems attractive is the Near Field Communication (NFC) technology (ISO/IEC Std. 18092 2004). It allows transmitting data simply by touching a smartphone to a receiving antenna. This can simplify payment processes and enable new applications (Ortiz Jr. 2006). However, the number of NFC enabled phones that are currently in use is too low to make it applicable today. According to the market research company Berg Insight, 30 million NFC enabled smartphones were sold in 2011 (Berg Insight AB 2011), while the total number of sold smartphones reached 468 million devices in 2011, according to (Gartner, Inc. 2011). Together with the still-in-use devices, of which almost none features NFC, the technology remains an interesting future option, but is not yet ready for reaching a large number of smartphone users today. Other wireless technologies, such as IEEE 802.11 (WIFI) (IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007) 2012) and Bluetooth (Bluetooth SIG 2010), are not suited for this task, since they require a complicated manual setup routine, which involves determining the correct network name and entering access codes. Such a routine would then have to be explained in detail to a large number of users and therefore increase the costs of such a solution drastically.

Our solution for transmitting digital receipts relies on barcode scanning technology, which is available in virtually every mobile phone and neither requires permanent Internet access nor additional hardware such as NFC-readers. Thereby, the proposed solution is widely applicable. Using built-in cameras to interact with products has already been evaluated, however until now this has mainly been done by using 1D barcodes. Karpischek and Michahelles (2010) implemented an application to scan product barcodes, to allow discussing these products online. Furthermore, they provide a customer with additional product information. Building up a database containing the products and the additional

information has been found to be vital to the success of such an application. One way to build it is by crowd-sourcing, i.e., letting the users build up the database themselves.

An approach that does exactly this was presented by Budde and Michahelles (2010), creating a game out of the data entry to motivate people to participate. Their stated goal was to create an open product information repository. This is different in our setting, where the data is supplied by the retailer. Also, the problems of scanning 1D barcodes with a mobile phone's camera, which have been observed by (von Reischach, Karpischek, Michahelles & Adelmann 2010), are not limiting our solution, since we rely on the already existing and tested scanning equipment present in current point-of-sale systems. This data is aggregated into a single 2D matrix code which has to be scanned by the customer, as opposed to scanning the 1D barcodes of all products. This is supported by observations made by (von Reischach, Karpischek, Michahelles & Adelmann 2010), who noticed that 2D matrix code scanning with mobile phone cameras has higher performance than the corresponding 1D barcode scanning technology.

Presently, as shown in Figure 1, retailers are already actively communicating with their suppliers and receive their bills via electronic means. For instance, between retailers and their suppliers, electronic billing is already available and in use. Often, these solutions are based on the EDIFACT standard (ISO Std. 9735 1990), the Electronic Data Interchange For Administration, Commerce and Transport. However, these solutions are tailored towards businesses, and are overly complex for the relationship that exists between retailers and their customers. Here, bills are currently presented in the form of paper-based receipts, for which we envision a digital replacement in order to extend the data of products from the supply chain beyond the point-of-sale.

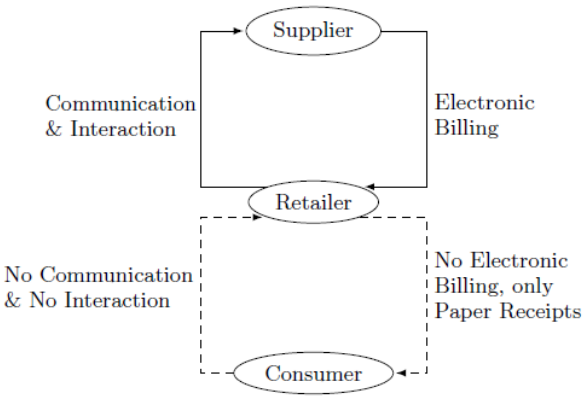

Figure 1. Current Relationships between Suppliers, Retailers and Customers

3 Encoding Receipts as QR-Codes

As mentioned in the introduction, digital receipts should be transmitted directly at the point of sale, to enable a direct interaction of the customer with his social peers based on the articles that he just bought[1]. Therefore, since we cannot assume that an Internet connection is always available at the point of sale, the receipt must be transmitted by other means. Since virtually any smartphone available these days is equipped with a camera, a barcode-based solution was devised. The barcode technology used are QR-codes (ISO/IEC Std. 18004 2006), due to their open availability and their relatively large data capacity, up to 2,953 Bytes, compared to other barcode technologies. Note however that the presented solution is not bound to this barcode technology; others, such as for example the DataMatrix code ((ISO/IEC Std. 16022 2006), could also be used instead.

Still, the amount of data that needs to be transmitted in case of a digital receipt quickly exceeds even the QR-code's capacity when done naively. Furthermore, recognizability of a QR-code is inversely proportional to the amount of data encoded, since finer patterns are required to encode more data. With patterns becoming finer, less space per data element is available, so that a camera reading the QR-code has less area, hence less pixels, available for each bit of information. Therefore, better alignment of the camera and more processing time is required for a smartphone to decode a QR-code

[1] Nb. Since the user decodes the barcode on his phone, the user can fully control the privacy and sharing settings.

containing lots of data. Also, when printed on paper, a finer QR-code becomes more damaged by wrinkles or stains. For these reasons, we devised an efficient compression scheme for digital receipts that is explained in the remainder of this section.

3.1 Encoding of Receipt Items

The key observation leading to the presented encoding scheme is that articles in a retail store are not sold equally often. Instead, there are articles that are sold in a large number of cases, and articles that are sold almost never. Thus, an article that is sold often should be encoded in a short way, whereas we can afford to spend more Bytes on seldom sold articles. A commonly used scheme for this kind of compression is the Huffman encoding (Huffman 1952), which is provably optimal, i.e., when encoding single items, no shorter encoding can be created (assuming that the probabilities of occurrence used to construct the encoding hold). Another property of this code is that it is prefix-free, which means that no encoding of one item can be extended to yield the encoding of another item. This will be of further use in our complete receipt encoding scheme, described later in this section. It must be noted that QR-codes already contain redundancy for better reliability. However, this redundancy is not aware of any redundancy of the data it encodes and therefore cannot be used in our algorithm. Hence, the idea behind our approach is to eliminate redundancy in the data to be able to add more redundancy in the QR-code that carries the receipt data. In cooperation with our partner from the retail sector, we first identified the data that needs to be transmitted on a digital receipt. These data fields are listed in Table 1.

Data Field Name	Example	Description
ID	7611700880705	Unique identification number (EAN/UPC or internal article number)
Scan name	Gerber Fondue	Name of the article shown on the receipt
Category	2352	Identifier of the category of articles to which this one belongs
Amount	3.00	Number of articles bought
Regular price	4.77	Price of the amount of articles that would be charged normally
Reduced price	3.63	Price of the amount of articles after deducing rebates

Table 1. Data fields comprising items

The identification number ID of an article uniquely determines this article. Therefore, this number is to be used when transmitting an item on a receipt. Often, this number is not shown on a paper receipt, however this number is the one that is being scanned at the point of sale and determines the remaining information (except for the amount and possible reductions that apply). A scan name of an article is the string displayed on a paper receipt. In our digital version, we use this name if no further information about an article is available, otherwise a more informative name is retrieved based on the ID of the article. Category information is included to enable article classification. This allows a customer to analyze his shopping based on the different types of articles bought. Such categories are identified by their number only, which can then be converted into a human-friendly representation by looking up this number in a pre-computed table.

How often a particular article is bought is of course specific to the current purchase, therefore it cannot be pre-determined and has to be included in the data to be transmitted. For a given retailer, this amount is always relative to some unit of measurement, e.g., pieces, ounces/grams, liters, etc. Such a unit of measurement is fixed for a given article, so it could be stored and does not have to be transmitted. However, it should be remarked that in current point-of-sale systems, units are often not available. Finally, the price of the amount of articles must be transmitted in the receipt data, too. This is the case, since prices of articles are frequently changed, to adapt to the current market situation. Furthermore, two separate prices are transmitted, because often rebates are available for certain articles, for example a price reduction due to a certain amount of articles being bought, and these are also changed frequently. Therefore, both the regular price and the reduced price, which is the price actually paid by

the customer, are included in the receipt data. By including both prices, we are able to inform the customer about his savings.

A receipt is then a collection of these receipt items, together with some global information about the purchase, such as date and time, store information, total amount, etc. Encoding the individual receipt items distinguishes two cases. In the first case, the information about the current article is already stored on the smartphone. In this case, it is not necessary to transmit all information, instead the ID of the article, the amount, and the two prices are sufficient. Furthermore, in this case the Huffman encoding discussed above can be used, by which the ID of the article is mapped to a number whose length reflects the frequency with which the article is bought. Thereby, top sellers that occur on lots of receipts take up the least space in the encoded receipt. Furthermore, the end of an encoded item is detected automatically, due to the fact that the employed Huffman encoding is prefix-free, as explained above. Thereby, as soon as an article with the given encoding has been found, no other article can be meant. This saves precious space, as the end of a variable-length encoding would otherwise have to be marked.

A summary of the fields contained in the encoding of items known on the smartphone is shown in Table 2, together with their size in bits and the resulting approximate maximal ranges. The amount is encoded as a fractional number with two decimal digits in 16 Bits, so that the possible range of numbers is from 0 to 655.35. Additionally, the two prices are contained, so that the data does not need to be updated whenever the price or the rebates change, which, as stated above, happens quite often. Also, only a fraction of the articles sold in a store have rebates applied to them. Hence, an extra bit before the two prices indicates whether a reduced price exists or not. In case no rebates are applied, the regular price is understood as the reduced price, and hence only needs to be encoded once. The length of the price encoding is given with 16/32 Bits. This allows distinguishing between small and large prices. Small prices are encoded in 16 Bits, whereas those exceeding the range of the small prices are encoded using 32 Bits. These two cases are distinguished by a special flag that is contained in the encoding: For small prices, the most significant bit is always set to 0, whereas for large prices, the most significant bit is always 1. Thereby, price ranges of -163.84 to 163.83 can be encoded in 16 Bits, while with 32 Bits prices between -10,737,418.24 and 10,737,418.23 can be encoded. Note that we also allow negative prices, to account for the fact that returned items, coupons, etc. also appear on receipts.

Data field	**Size**	**Approximate Range**
Huffman code	*variable size*	
Amount in 1/100	16 Bits	0 – ca. 650
Reduced price flag	1 Bit	0 / 1
Regular price	16/32 Bits	Ca. -10m – ca. 10m
Reduced price	16/32 Bits	Ca. -10m – ca. 10m

Table 2. Data encoding of known items

Data field	**Size**	**Approximate Range**
Article ID	*44 Bits*	0 – ca. 1.7×10^{13}
Amount in 1/100	16 Bits	0 – ca. 650
Reduced price flag	1 Bit	0 / 1
Regular price	16/32 Bits	Ca. -10m – ca. 10m
Reduced price	16/32 Bits	Ca. -10m – ca. 10m
Category	14 Bits	0 – ca. 16,000
Scan name	Max. 160 Bits	Up to 20 letters

Table 3. Encoding of articles not known to the smartphone

A different encoding is chosen when the article on the receipt is not known to the smartphone. This for example happens when new articles are introduced and the database on the smartphone has not yet been updated, or when an article is being bought that normally is sold so infrequently that it does not

make sense to store it on the restricted space available on a smartphone. In this case, the information shown in Table 1 needs to be encoded in a different way. Since the lookup of Huffman codes is not possible (as the article is not known to the smartphone), the full article ID needs to be transmitted. Furthermore, a scan name should also be transmitted, to display to the user rather than the incomprehensible article ID. Also, to be able to analyze category data, the article's category should be transmitted. Finally, the two different prices must be transmitted. This data is summarized in Table 3.

In the article ID, either a universal product code (UPC) / international article number (EAN) (i.e., a Global Trade Item Number (GTIN) (GS1 US 2006) without packaging indicator digit) is encoded, or an internal article ID. A UPC/EAN code uniquely identifies a product globally, and is the number printed as a 1D barcode on the product. These numbers are up to 13 digits in length. Internal article IDs are only valid for a single retailer. To encode them, the two most significant bits in the article ID data field are set to 1, thereby leaving the allowed range of UPC/EAN codes so that it can be uniquely determined whether a UPC/EAN code or an internal article number is contained. Using this scheme allows a total of ca. 4 trillion internal article IDs. The amount and the up to two prices are encoded as in the case of articles known to the smartphone. Finally, the category ID and the displayed scan name are encoded, with the latter being up to 20 letters in length. If less letters are used, then the scan name is terminated with 8 bits all set to 0 (as is standard practice in numerous programming languages).

Note that the above encoding is only included as a "last- resort", i.e., only for those cases where information about an article cannot be stored on the smartphone. Hence, while still important, this encoding is expected to be used in only a small number of cases.

3.2 Creating QR-Code Receipts

Using the encoding presented above, we can encode complete receipts in a QR-Code that is easily readable by cameras contained in current smartphones. As already noted, a receipt is a collection of individual items, together with some global information. Hence, our encoding of complete receipts reflects this. The data fields of such a receipt are listed in Table 4. The version field is included to allow future updates of the format. Padding denotes a field that indicates the number of bits that were added to the end of the data stream to obtain a multiple of 8. This is needed since QR-codes only represent Bytes, with each Byte consisting of 8 bits. Each receipt in a store is assigned a unique number that is recorded in the field Receipt ID. Usually, this includes the market where the receipt was issued, otherwise this has to be added as an extra field. The last of the global information fields is the timestamp, which records the date and time of the receipt. This is followed by an enumeration of the individual items encoded using either of the two encodings presented previously. As mentioned there, usually the encoding for items known on the smartphone is used, however unknown items may always appear. To distinguish between these two encodings, each entry is preceded by a type indicator that determines the encoding used.

Version	Padding	Receipt ID	Timestamp	$Item_1$	…	$Item_N$

Table 4. Data fields of a complete receipt

An example of such a receipt is shown in Figure 2, together with a break down into the individual bits comprising the different fields. Note that the QR-code can be read with any available reader software, however the contained data needs to be decoded according to the above scheme to result in a digital receipt. A prototype of such a mobile application has been developed for the present evaluation.

In Figure 2, bits are represented by a dot '.' for an unset and an 'X' for a set bit. The version of the receipt is 0 (currently the only version number defined) and the padding is 3, i.e., 3 bits needed to be appended to obtain a total size that is evenly divisible by 8. The ID of the receipt is stored in the next 64 bits and comprises the market ID, date, checkout ID, and a running receipt number. Converted into decimal, it reads 42-20120509-03-0732, with dashes added to indicate the boundaries of the

components. In the timestamp, an unsigned 32 bit representation of seconds since January 1st, 1970 at 0:00 UTC is stored, a common representation of time in computer systems. In the example, the timestamp represents the date “Wed May 9 12:50:27 UTC 2012”. Afterwards, the individual items are encoded. This receipt contains two items from our test database.

Version:	....
Padding:	.XX
Receipt ID:	.....XXX.XXXXXXX...X.XX.X
	.X.XXXXXXX.XX..XX.X....X.X..XX..
Timestamp:	.X..XXXXX.X.X.X..XX.X.....X..XX
Item 1	
Type:	.X
Huffman code:	X...X.
Amount:	.....XX..X...
Flag:	X
Regular Price:	....X..X.X..XX..
Reduced Price:	.....XX.XXXXX..X
Item 2	
Type:	..
Article ID:	.XXXX.X..XX...X.XX.X..
	.X..XX...XX.XX.X.X.XXX
Amount:	.....XX..X..
Flag:	.
Regular Price:	.....XX.XXXX.X.
Category:	...X..XX.X..X.
Scan Name:	“Rioja Campo Viero\0”
Padding:	...

Figure 2. QR-Code of a receipt encoding 2 items

The first item is known on the mobile, indicated by the type “.X” (i.e., 1). Thus, using a short Huffman code suffices to look it up in the database. It has amount 2.00, i.e., this article is bought twice, and a 50% off reduction when bought more than once applies. Thus, not the regular price of 23.80 has to be paid, but only 17.85, which is encoded in the two price fields. Presence of the second price is indicated by the flag being set.

For demonstration purposes, we pretended that the second item is not contained in the database on the mobile device. Hence, the encoding for unknown items is used, which is indicated by the type being 0. Following the type is the article identifier, which in this case is the EAN 8410302192471 since the highest bit is unset. The amount is 1.00 and no reductions apply. Hence, the flag is 0 and only the regular price is encoded. Following is the category identifier 1234 and a short scan name. In this case, the scan name is only 17 characters long, hence it is terminated with an 18th character of eight unset bits, indicated with “\0” in the figure.

Global Data	Item 1	Item 2

Figure 3. Proportions of the data in the receipt encoding of Figure 2

In total, the receipt is encoded using 50 Bytes, of which 13.25 Bytes are spent on the global data (including the padding). The known item is encoded using 7.125 Bytes, while for the unknown item 29.625 Bytes are required. The proportions of the data sizes are also illustrated in Figure 3, which clearly demonstrates why the encoding of known items is to be preferred, since in this example the encoding of the unknown item is ca. 4 times larger than the encoding of the known item. Note that the main source of the blow-up is the scan name, which in this example takes up 18 Bytes, i.e., more than 60 % of the item’s encoding. Thus, if lots of unknown items have to be encoded, a possible option is to shorten the displayed scan names. However, we want to stress again that the main mode of

operation is using the encoding of known items, which is achieved by updating the database on the smartphone regularly.

When only encoding known items, then a QR-code can easily hold large receipts. As an example, Figure 4 shows a QR-Code containing a receipt with 200 different items, each having an arbitrary amount. This receipt has a total size of 989 Bytes and its corresponding QR-Code is still quickly decoded by the ZXing project's barcode scanning library (ZXing 2012) used in our prototype implementation.

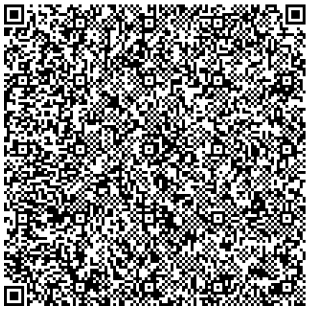

Figure 4. QR-Code containing a receipt of 200 items

4 Experimental Evaluation

To assess the efficiency of our compression scheme, which we call FBRC for frequency-based receipt compression, we compared it against well-known general purpose compression algorithms. For our comparisons, we chose the wide-spread Deflate algorithm (Storer & Szymanski 1982, Salomon 2007) which is the de-facto standard of the ZIP compression format, and the LZMA algorithm (Pavlov 2012, Salomon 2007), a modern and powerful compression format that is becoming more and more popular (Lindholm 2009).

In the below experiments, we compare the original Byte size, the size in Bytes of our encoding for known articles, and the Byte sizes after applying the Deflate and the LZMA algorithms on the original input data. In a first experiment, we randomly generated 1,000 receipts, where the probability of an article occurring on a receipt was proportional to the frequency with which it occurred in a data set of over 200,000 real-world receipts kindly provided by our retail partner. Each receipt had exactly 13 items on it, which is the mean number of items per receipt in the real-world data set. A graphical representation of the results can be seen in Figure 5, where the x-axis displays the different receipts and the y-axis shows the size of the receipt in Bytes. In the figure, we denote with "Plain" the simple enumeration of all article IDs, the amount bought, and the price. This number is to serve as a baseline against which we compare the other options. The line labeled "FBRC" represents the encoding of known articles presented in Section 3, while "Deflate" and "LZMA" represent the result of compressing the plain data with the respective algorithm.

Figure 5 shows that FBRC clearly outperforms the two general-purpose compression algorithms on these generated receipts. Also, it can be observed that FBRC reduced the amount of data needed to be transmitted roughly into half. On average, it only needs 49.39 % of the plain size, while the Deflate and LZMA compressed receipts take up 92.83 % and 89.25 % of the plain size, respectively.

To help the general-purpose compression schemes, we increased the number of items on the randomly generated receipts to 50. This allows the general purpose compression algorithms to learn more about the distribution, thereby allowing them to reduce the data size further. Also, this is approximately the maximum number of different items that occur on a private shopper's receipt. However, our FBRC encoding scheme still outperforms the general-purpose ones significantly, as can be seen from Figure 6. As an interesting aside, this figure also shows that Deflate and LZMA correlate, due to their common roots in the Lempel Ziv algorithm (Lempel & Ziv 1977, Salomon 2007).

Our target application are however real receipts which have varying numbers of items. From our real-world data set, we randomly selected 1000 receipts and, like before, encoded them with the four schemes considered here: as plain enumeration, using the FBRC scheme from Section 3 for known articles, and using the two compression algorithms Deflate and LZMA. The results for these receipts are shown graphically in Figure 7. Note that the size on the y-axis has been cut off at 500, the plain enumeration of some receipts took up to 900 Bytes. This occurred for receipts with lots of items on them. In the data set considered for the experiment, the number of items on a receipt ranged from only 1 item on a receipt to 75 items.

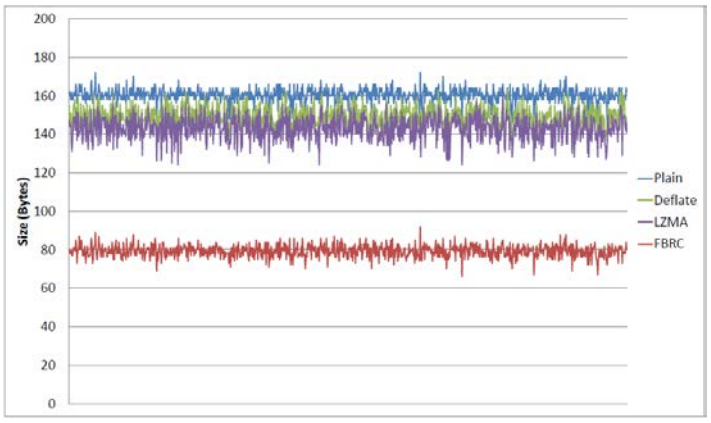

Figure 5. Results for randomly generated receipts with 13 lines

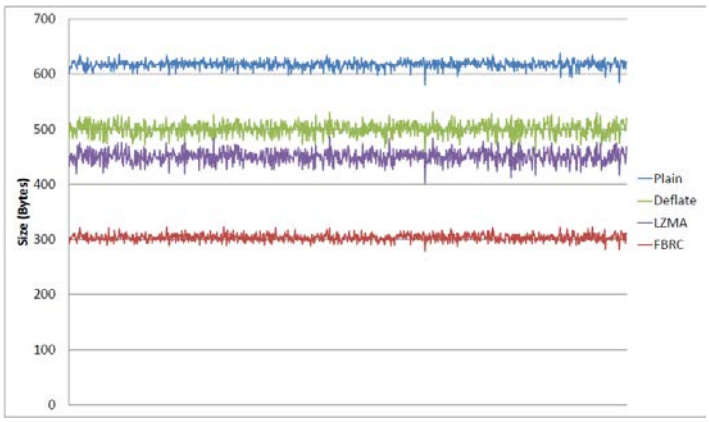

Figure 6. Results for randomly generated receipts with 50 items

The results show that the FBRC encoding scheme reduces the size of an encoded receipt to a small percentage of the original size for our real-world receipts. Expressed in numbers, an FBRC encoded receipt uses only 48.30% of the plain enumeration size on average, whereas Deflate and LZMA average to 93.40% and 101.86% of the original size, respectively. Especially the last number is surprising, but can be explained by the large number of small receipts, where both of the compression algorithms result in an increase in size. This is never the case for FBRC, which always stays below the size requirements of the plain enumeration in our experiment. When computing the averages of the sizes, we get that the mean of the plain enumeration takes 131.5 Bytes, while FBRC takes up 63.27 Bytes on average. The two general-purpose compression algorithms have means of 105.76 Bytes for Deflate and 105.31 Bytes for LZMA, which shows that for larger receipts these algorithms can do their work and reduce the space required, but still are beat by our domain-specific encoding.

On the individual receipt level, the experiment's data shows that FBRC is always better than either of the two compression algorithms except for two cases in which it is beat by LZMA: In the first case, the output of LZMA is 1 Byte less, but in the other case the difference is larger, namely 332 Bytes for LZMA vs. 359 Bytes for FBRC. This last case was investigated further, by retrieving the receipt from the data set. The receipt in question was a rather large one, containing a total of 61 items. However, a large number of these items were referring to the same article that was bought, but were listed separately on the receipt. This explains why the LZMA compression algorithm was so successful on this particular receipt: It does not treat items individually (since it does not know about this concept) and therefore can make use of this redundancy. FBRC however treats these items separately, encoding each item on its own. Hence, we experimented with a reduced version of the receipt, in which the items referring to the same article were collapsed into a single item, with properly adjusted amounts and prices. This version of the receipt only consists of 24 items, and takes up 288 Bytes in the plain enumeration, 252 and 236 Bytes when compressed with Deflate and LZMA, respectively, and only

141 Bytes using FBRC. Hence, pre-processing the receipts to reduce such redundant items results in smaller receipt sizes and makes our encoding perform much better than the general-purpose compression algorithms. Applying this pre-processing step can easily be done before encoding the receipt.

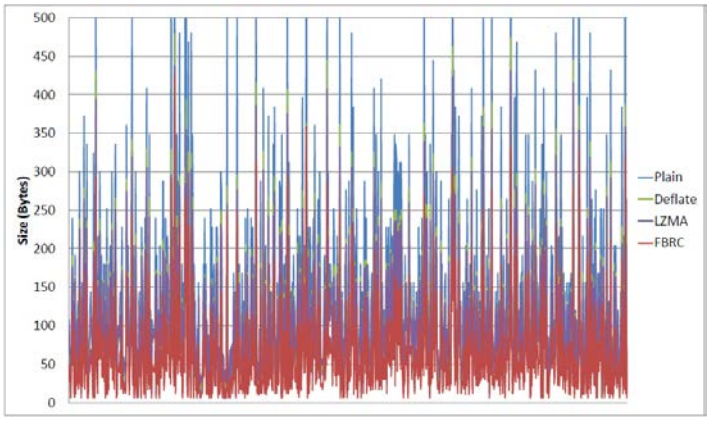

Figure 7. Results of encoding real world receipts

Finally, we ran the comparison between plain size, FBRC, Deflate, and LZMA on the full data set consisting of 209,689 real-world receipts. Prior to encoding a receipt, we pre-processed it as explained above, to remove redundancies. Then, the receipt was encoded with the four different approaches. The results are shown in Table 5 and depicted graphically in Figure 8. Note that these results only show average sizes; the maximum size of a receipt was much larger due to the very different sizes of real-world receipts (as already observed in the subset considered for Figure 7).

Encoding	Average Size	Standard Deviation
Plain	127.27 Bytes	113.61 Bytes
Deflate	110.05 Bytes	85.80 Bytes
LZMA	109.40 Bytes	77.14 Bytes
FBRC	66.76 Bytes	61.18 Bytes

Table 5. Results for the real-world data set with pre-processing

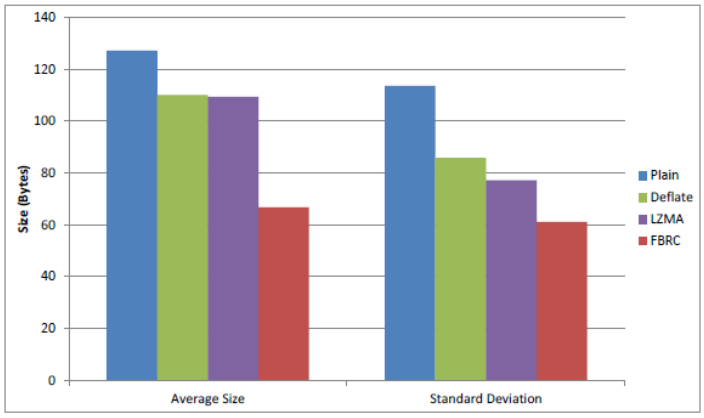

Figure 8. Graphical representation of the results for real-world receipts shown in Table 5

On average, a receipt took up 127.27 Bytes when plainly enumerating items. With Deflate and LZMA, the average sizes were 110.05 and 109.40 Bytes, respectively. Again, this shows that due to numerous small receipts the compression algorithms cannot reduce the size a lot. This is different with FBRC, with an average size of 66.76 Bytes to encode a full receipt consisting of the articles bought, their amounts, and prices. Also, the standard deviation is smaller using FBRC, which is consistent with the

graphics in Figures 5 and 6. There, the amplitude of FBRC is smaller than that of any other encoding scheme. Thus, FBRC is also better predictable, roughly reducing the space required by one half.

5 Conclusion and Possible Uses of Digital Receipts

In this paper, the domain-specific encoding of receipt data called FBRC was presented. It makes use of the known relative frequency with which an article is bought, which allows to easily transmit receipts from the point-of-sale to a smartphone using QR-codes, a 2D matrix code technology that is attractive due to the fact that virtually every smartphone is equipped with a camera. Thereby, bringing the physical goods being sold into the virtual Internet of things is not hampered by technological hurdles.

In case a new article needs to be added or the frequency of an existing one changes, then this change needs to be propagated to both the barcode generator as well as the smartphone. From this information, a new Huffman coding can be built and the encoding continues to work. Until then, a transition phase has to be entered during which the encoding for unknown articles has to be used in case of new articles, or the encoding is not as efficient as it could be when the frequency of an article changed by a large amount. In our experiments, we have demonstrated that using domain-specific knowledge is far superior to employing off-the-shelf compression algorithms that work well in other, less predictable settings. Especially, our encoding scheme FBRC was able to constantly reduce the size as compared to the original input data size on receipt data that reflects the usual shopping behavior in retail stores. This was not the case for the compression algorithms that need larger input data to be able to work.

On the real-world receipts, which are our target application, FBRC also outperforms the general-purpose compression algorithms. Especially, it never occurred in our experiments that this encoding increased the resulting size, which happened with the compression algorithms. Still, we found two instances out of 1,000 real-world, raw receipts considered, for which the general-purpose compression algorithms worked better than the FBRC encoding scheme. This was due to some redundancy present in today's receipt data, where multiple entries on a receipt are referring to a single product. By adding a simple pre-processing step we were able to reduce the size of the encoded receipts for all considered encoding schemes. On the pre-processed data, FBRC again performs better than the generic compression schemes, which was evaluated for a large set of real-world receipts. It should be noted that the pre-processing step will never increase the size of the resulting encoded receipt, the only disadvantage is the small computation overhead it incurs. However, both the pre-processing and the encoding can be done in negligible time on modern computer hardware. Our solution is of course limited by the amount of data that QR-codes can contain on the one side, and the capability of decoding dense QR-codes in a smartphone on the other side. However, as our extensive experiments have proven, our proposed algorithm has sufficient capability to deal with real-world receipts.

As we explained in Section 2, our solution does not depend on hardware features of smartphones that are currently hardly available, such as the NFC technology. In case such a technology should gain wide-spread acceptance, then it becomes a main competitor to the proposed barcode scanning technique. Note however that for all data transmissions, reducing the amount of plain data is always beneficial, as it reduces the time needed for transmission and/or gives room for additional error mitigation. Therefore, the proposed algorithm could also be of in such environments.

As already mentioned in the introduction, digital receipts allow numerous applications. For example, they enable customers to track their expenses on their mobile phones, compare their purchase patterns with their desired behavior (such as, for example, buying biologically produced goods), and communicating with peers about products bought. Also retailers and brands profit from digital receipts. They can, for example, get direct feedback on purchases and the choices that lead to them. Also, digital receipts allow a continuous communication with the customers, instead of only pushing out information (such as flyers, emails, etc.). All of this needs a simple and robust means of transferring receipts digitally, for which we presented a domain-specific encoding to be able to fit a complete receipt into a single QR-code that is easy to read with current smartphones.

References

Berg Insight AB (2011), ‘News 2012-03-26: Shipments of NFC-enabled handsets reached 30 million units in 2011’. http://www.berginsight.com Last accessed 2012-05-09.

Bluetooth SIG (2010), ‘Bluetooth Specification Version 4.0’.

Budde, A. & F. Michahelles (2010), Product Empire – Serious play with barcodes, in ‘Internet of Things (IOT 2010)’, pp. 1–7.

Dellarocas, C. (2003), ‘The Digitalization of Word of Mouth: Promise and Chal- lenges of Online Feedback Mechanisms’, Management Science 49(10), 1407–1424.

Gartner, Inc. (2011), ‘Press Release April 7, 2011: Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012’. http://www.gartner.com/it/page.jsp?id=1622614 Last accessed 2012-05-09.

Google Inc., Otto Group, TNS Infratest, and Trendbüro (2012), ‘Go Smart2012: Always-in-Touch’. http://www.ottogroup.com/media/docs/de/studien/go_smart.pdf Last accessed 2012-05-09.

GS1 US (2006), ‘An Introduction to the Global Trade Item Number (GTIN)’.

Huffman, D. A. (1952), A Method for the Construction of Minimum-Redundancy Codes, in ‘Proceedings of the I.R.E.’, pp. 1098–1101.

IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007) (2012), ‘IEEE Stan- dard for Information technology–Telecommunications and information ex- change between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications’.

ISO Std. 9735 (1990), ‘Electronic data interchange for administration, commerce and transport (EDIFACT) – Application level syntax rules’.

ISO/IEC Std. 16022 (2006), ‘Information technology – Automatic identification and data capture techniques – Data Matrix bar code symbology specification’.

ISO/IEC Std. 18004 (2006), ‘Information technology – Automatic identification and data capture techniques – QR Code 2005 bar code symbology specification’.

ISO/IEC Std. 18092 (2004), ‘Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1)’.

Karpischek, S. & F. Michahelles (2010), my2cents – digitizing consumer opinions and comments about retail products, in ‘Internet of Things (IOT 2010)’, pp. 1–7.

Lempel, A. & J. Ziv (1977), ‘A Universal Algorithm for Sequential Data Compression’, IEEE Transactions on Information Theory 23(3), 337–343.

Lindholm, B. (2009), ‘New Options in the World of File Compression’, Linux Gazette (162).

Ortiz Jr., S. (2006), ‘Is Near-Field Communication Close to Success?’, Computer 39(3), 18–20.

Pavlov, I. (2012), ‘LZMA SDK’. http://www.7-zip.org/sdk.html version 9.20. Last accessed 2012-05-09.

Salomon, D. (2007), Data Compression: The Complete Reference, 4th edn, Springer-Verlag.

Senecal, S. & J. Nantel (2004), ‘The Influence of Online Product Recommendations on Consumers’ Online Choices’, Journal of Retailing 80(2), 159–169.

Storer, James A. & Thomas G. Szymanski (1982), ‘Data compression via textual substitution’, J. ACM 29(4), 928–951.

von Reischach, F., E. Dubach, F. Michahelles & A. Schmidt (2010), An Evaluation of Product Review Modalities for Mobile Phones, in ‘Proceedings of the 12th ACM SIGCHI SIGMOBILE International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI’10)’.

von Reischach, F., S. Karpischek, F. Michahelles & R. Adelmann (2010), Evaluation of 1D barcode scanning on mobile phones, in ‘Internet of Things (IOT 2010)’, pp. 1–5.

ZXing (2012), ‘Multi-format 1D/2D barcode image processing library with clients for Android, Java’. http://code.google.com/p/zxing Last accessed 2012-05-09.