Security Issues for 2D Barcodes Ticketing Systems

Cristian Toma

Cybernetics and Economic Informatics Faculty
Academy of Economic Studies
Romana Square 6, Bucharest
ROMANIA
cristian.toma@ie.ase.ro

Abstract: The paper presents a solution for endcoding/decoding access to the subway public transportation systems. First part of the paper is dedicated through section one and two to the most used 2D barcodes used in the market – QR and DataMatrix. The sample for DataMatrix is author propietary and the QR sample is from the QR standard [2]. The section three presents MMS and Digital Rights Management topics used for issuing the 2D barcodes tickets. The second part of the paper, starting with section four shows the architecture of Subway Ticketing Systems and the proposed procedure for the ticket issuing. The conclusions identify trends of the security topics in the public transportation systems.

Key-Words: 2D barcode, Ticketing System, DataMatrix code – ISO/IEC 16022, QR code – ISO/IEC 18004, MMS – Multimedia Message Service, M-DRM – Mobile Digital Rights Management , 2D Barcode Automatic Ticketing System – 2D BATS

1 DataMatrix – ISO/IEC 16022

Parts from this section are copyrighted by ISO/IEC 16022 [1]. Data Matrix is a 2D matrix symbology. According with [1], there are 2 types of symbologies:

- ECC 200 which uses Reed-Solomon error correction. ECC 200 is recommended for new applications.
- ECC 000 - 140 with some levels of convolutional error correction, referred to as ECC 000, ECC050, ECC 080, ECC 100 and ECC 140 respectively. ECC 000 - 140 should only be used in closed applications where a single party controls both the production and reading of the symbols and is responsible for overall system encoding/decoding procedures.

The characteristics of Data Matrix are [1]:

- Encodable character set:
  - values 0 - 127 in accordance with ISO/IEC 646, i.e. all 128 ASCII characters (equivalent to the U.S. national version of ISO 646)
  - Values 128 - 255 in accordance with ISO 8859-1. These are referred to as extended ASCII.
  - Representation of data: A dark module is a binary one and a light module is a zero.
- Symbol size in modules (not including quiet zone):
  - ECC 200 is for 10 x 10 to 144 x 144 even values only
  - ECC 000 – 140 is for 9 x 9 to 49 x 49, odd values only
- Data characters per symbol (for maximum symbol size in ECC200):
  - Alphanumeric data: up to 2335 characters
  - 8-bit byte data: 1555 characters
  - Numeric data: 3116 digits.
- Selectable error correction:
  - ECC 200: Reed-Solomon error correction.
  - ECC 000 - 140: Four levels of convolutional error correction, plus the option to apply only error detection
- Code type: 2D Matrix
- Orientationindependence: Yes

ECC 200 alignment patterns for 32x32 square symbol – figure 1.1:

ISSN 2067 – 4074

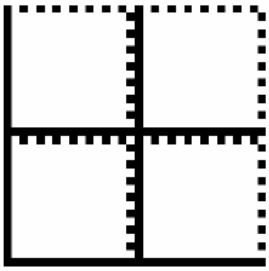


Figure 1.1 Datamatrix 32x32 square symbol

The sample from the Table 1.1 is author property and is NOT a replacement of the standard [1]. For a complete reference please checkout ISO store for ISO/IEC 16022 standard [1]. Starting from the sample from Annex O (informative) [1], there is a description of another encodation type at byte level for word *"Ana"*:

Table 1.1 ECC200 Paper Sample

Step 1: Encoding

The ASCII representation is:
Data character: 'A' 'n' 'a'
Decimal: 65+1 110+1 97+1
Hex: 0x42 0x6F 0x62

ASCII encodation is the default set for the first symbol character in all symbol sizes. It encodes ASCII data, double density numeric data and symbology control characters. Symbology control characters include function characters, the pad character and the switches to other code sets. ASCII data is encoded as codewords 1 to 128 (***ASCII value + 1***).

Consulting Table 7 from [1], 3 data codewords fit exactly into a 10 x 10 symbol, and five error correction codewords need to be added. If the encoded data did not exactly fill a data region, then additional pads would have to be encoded.

Table 7 — ECC 200 symbol attributes

Symbol size[a]		Data region		Mapping matrix size	Total codewords		Reed-Solomon block		Inter-leaved blocks	Maximum data capacity		
Row	Col	Size	No.		Data	Error	Data	Error		Num.	Alphanum.[4]	Byte
10	10	8 x 8	1	8 x 8	3	5	3	5	1	6	3	1
12	12	10 x 10	1	10 x 10	5	7	5	7	1	10	6	3
14	14	12 x 12	1	12 x 12	8	10	8	10	1	16	10	6

Partial sections from Table 2 from [1] are relevant for ASCII values encoding:

Table 2 — ASCII encodation values

Codeword	Data or function
1 - 128	ASCII data (ASCII value + 1)
129	Pad
130 - 229	2-digit data 00 - 99 (Numeric Value + 130)

Step 2: Error checking and correction

Error correction codewords are generated using the Reed-Solomon algorithm and appended to the encodation data stream – keep in mind, that the one should add 1 for each codeword in data, in order to obtain the ASCII encoding and the proper Reed-Solomon correction codes.

The resulting data stream is:
Data: Ana
CW No: 1 2 3 4 5 6 7 8
decimal: 66 111 98 20 66 57 160 115
hex: 42 6F 62 14 42 39 A0 73
_____data_____/
_____check_____/

Annex E in [1] describes the error correction process for ECC 200 and Annex E.3 in [1] gives an example of a routine to perform the calculation of the error correction codewords for the data "123456".

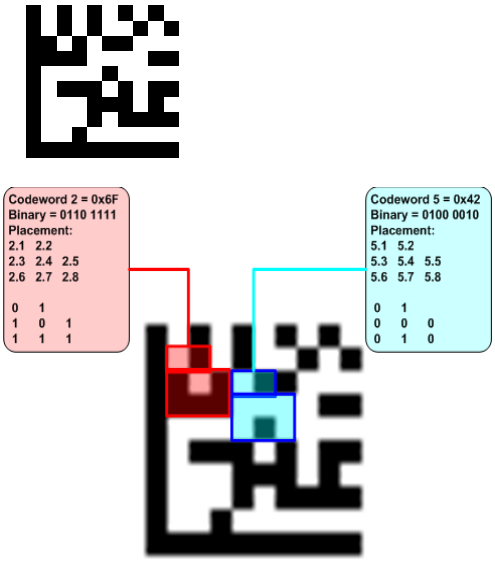
Step 3: Module placement in matrix

The final codewords from Step 2 are placed in the binary matrix as symbol characters according to the algorithm described in [1] section 5.8.1 (also see Figure F.1 and F.9 from [1]) - Where 2.1

is the first bit, 2.2 is the second bit from codeword 2, and so on:

Step 4: Actual symbol

The final Data Matrix symbol is produced by adding the finder pattern modules and converting the binary ones to black and binary zeroes to white. For instance, the second byte has value 0x6F = 0110 1111 and the fitfh codeword has value 0x42 = 0100 0010, the following figure highlights this issue:



3. QR – ISO 18004

Parts from this section are copyrighted by ISO/IEC 18004 [1]. QR – Quick Response is a 2D matrix symbology invented by Toyota, subsidiary Denso-Wave in 1994. The QR code is one of the most popular types of two-dimensional barcodes.

The characteristics of QR are [2]:

- Encodable character set:
  - numeric data (digits 0 - 9);
  - alphanumeric data (digits 0 - 9; upper case letters A -Z; nine other characters: space, $ % * + - . / :);
  - 8-bit byte data (JIS 8-bit character set (Latin and Kana) in accordance with JIS X 0201);
  - Kanji characters (Shift JIS character set in accordance with JIS X 0208 Annex 1 Shift Coded Representation.)
- Representation of data: A dark module is a binary one and a light module is a binary zero.
- Symbol size (not including quiet zone): 21 x 21 modules to 177 x 177 modules (Versions 1 to 40, increasing in steps of 4 modules per side)
- Data characters per symbol (for maximum symbol size – Version 40-L):
  - numeric data: 7089 characters
  - alphanumeric data: 4296 characters
  - 8-bit byte data: 2953 characters
  - Kanji data: 1817 characters
- Selectable error correction: Four levels of error correction allowing recovery of the symbol codewords:
  - L 7%
  - M 15%
  - Q 25%
  - H 30%
- Code type: 2D Matrix
- Orientation independence: Yes
- The structure of QR code is in figure 2.1 and 2.2 – copyright [2] and [8]:

ISSN 2067 – 4074

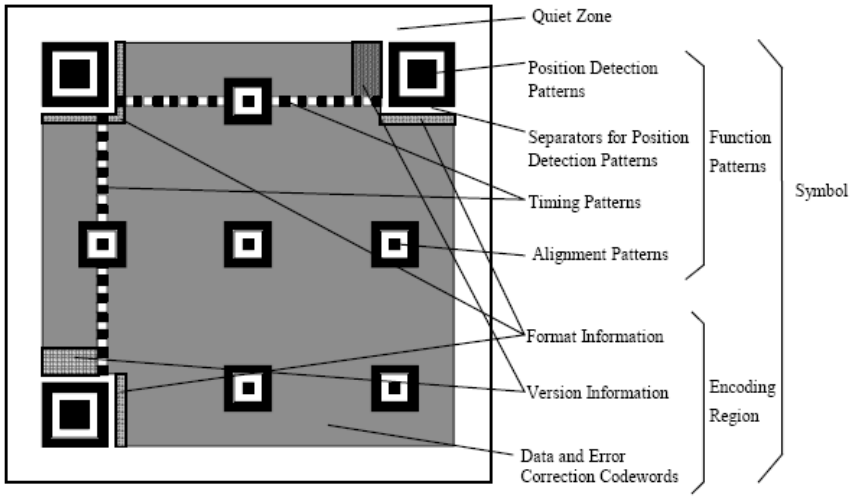


Figure 2.1 QR Structure from standard [2]

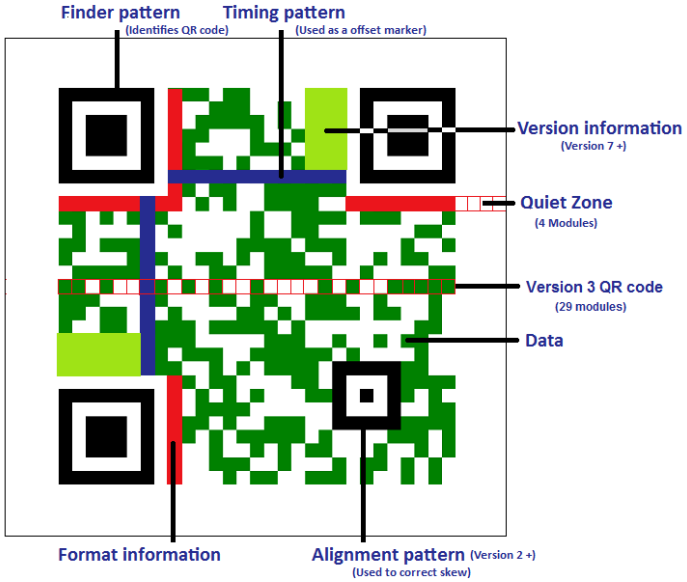


Figure 2.2 QR Structure [8]

The sample from the Table 2.1 is NOT author property and is NOT a replacement of the standard [2]. For a complete reference please checkout ISO store for ISO/IEC 18004 standard [2]. Starting from the sample from Annex G (informative) [2], there is a description of the encodation type at numeric level for data string *"01234567"*:

This section describes the encoding of the data string 01234567 into a version 1-M symbol, using the Numeric Mode in accordance with 8.4.2. from [2].

Step 1: Data Encodation
- Divide into groups of three digits and convert each group to its 10 or 7 bit binary equivalent:
012 -> 0000001100
345 -> 0101011001
67 -> 1000011

Step 1.1 - Convert Character Count Indicator to binary (10 bits for version 1-M – is between version 1 to 9 and has correction level M) => Character count indicator (value 8 digits for data string "*01234567*") = 0000001000 – see Table 2 and 3 from standard [2]:

Table 2 — Mode indicators

Mode	Indicator
ECI	0111
Numeric	0001
Alphanumeric	0010
8-bit Byte	0100
Kanji	1000
Structured Append	0011
FNC1	0101 (First position) 1001 (Second position)
Terminator (End of Message)	0000

Table 3 — Number of bits in Character Count Indicator

Version	Numeric Mode	Alphanumeric Mode	8-bit Byte Mode	Kanji Mode
1 to 9	10	9	8	8
10 to 26	12	11	16	10
27 to 40	14	13	16	12

source: [2]

The end of the data in the complete symbol is indicated by a 4 bit terminator **0000**, which is omitted or abbreviated if the remaining symbol capacity after the data bit stream is less than 4 bits. The terminator is not a Mode Indicator as such.

Step 1.2 - Connect Mode Indicator for Numeric Mode (**0001**), Character Count Indicator, binary data, and Terminator (**0000**)
0001 0000001000 0000001100 0101011001 1000011 0000

Step 1.3 - Divide into 8-bit codewords, adding padding bits (shown underlined for illustration) as needed
00010000 00100000 00001100 01010110 01100001 10000000

Step 1.4 - Add Pad codewords to fill data codeword capacity of symbol (for version 1-M, 16 data codewords, therefore 10 Pad codewords required (shown underlined for illustration)), giving the result – see table 7 from standard [2]:

Table 7. Number of symbol characters and input data capcity for version 1 to 8 (source [2])

Version	Error correction level	Number of data codewords[a]	Number of data bits[b]	Data capacity			
				Numeric	Alphanumeric	8-bit Byte	Kanji
1	L	19	152	41	25	17	10
	M	16	128	34	20	14	8
	Q	13	104	27	16	11	7
	H	9	72	17	10	7	4

00010000 00100000 00001100 01010110 01100001 10000000 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001

The padding is established in section 8.4.9 from standard [2] where "The message bit stream shall then be extended to fill the data capacity of the symbol corresponding to the Version and Error Correction Level, as defined in Tables 7 to 11, by the addition of the Pad Codewords **11101100** and **00010001** alternately".

Step 2: Error Correction Codeword generation

ISSN 2067 – 4074

Using the Reed-Solomon algorithm to generate the required number of error correction codewords (for a Version 1-M symbol, 10 are needed – see table 13 from standard [2]), these (shown underlined for illustration) should be added to the bit stream, resulting in:

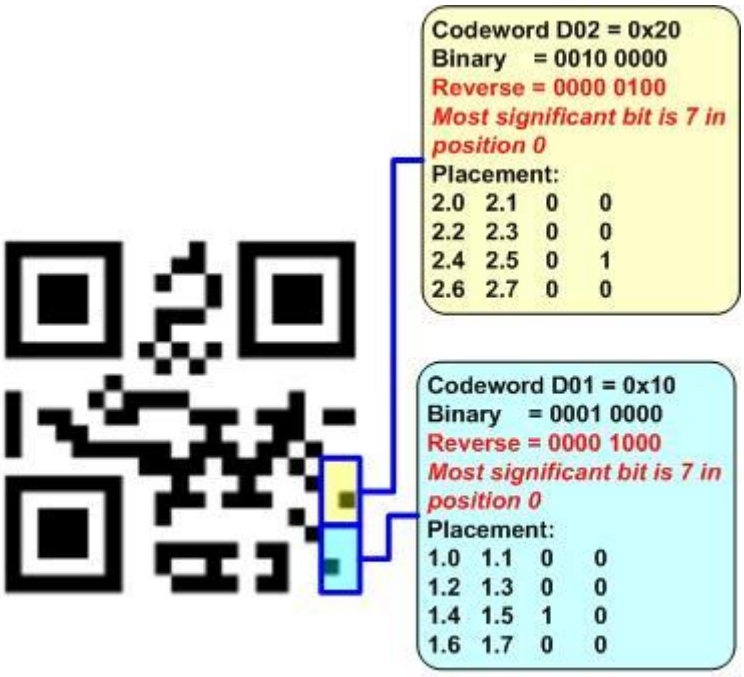
00010000 00100000 00001100 01010110 01100001 10000000 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 <u>10100101 00100100 11010100 11000001 11101101 00110110 11000111 10000111 00101100 01010101</u>

Table 13 — Error correction characteristics for versions 1 to 6

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
1	26	L	7	1	(26,19,2) [b]
		M	10	1	(26,16,4) [b]
		Q	13	1	(26,13,6) [b]
		H	17	1	(26,9,8) [b]

Step 3: Module placement in matrix

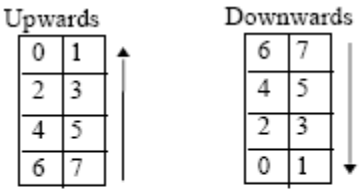
As there is only a single error correction block in a version 1-M symbol, no interleaving is required in this instance. The Position Detection Patterns and Timing Patterns are placed in a blank 21 x 21 matrix (26 Data and Error Correction Codewords) and the module positions for the Format Information are left temporarily blank. The codewords from Step 2 are placed in the matrix in accordance with 8.7.3. from standard [2] – figure represents data modules prior the masking process – version 1-M:



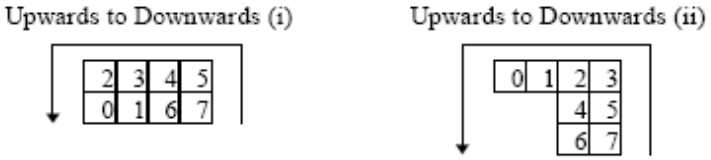
The most significant bit (shown as bit 7) of each codeword shall be placed in the first available module position.

The figure copyrighted by author for this paper

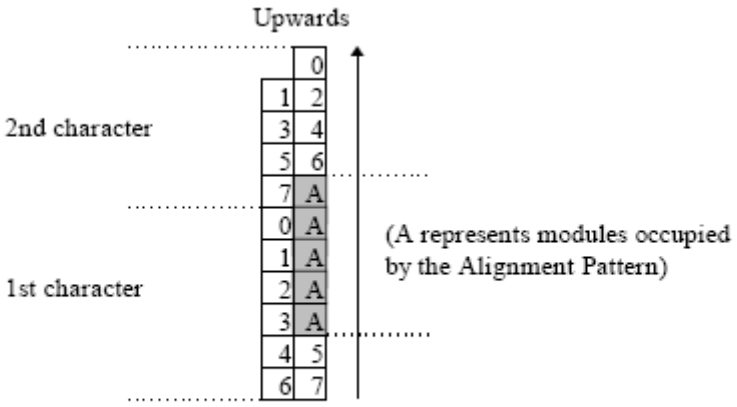
For bits placements in Data and Error Correction Codewords please see figures 12, 13, 14 and 15 from the standard [2]:



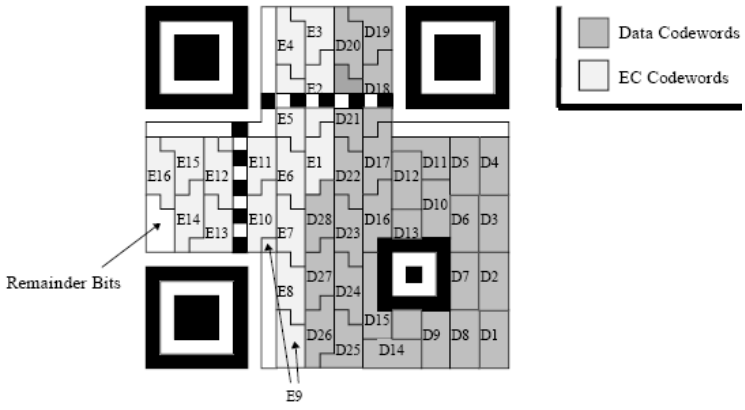
Bit placement in regular symbol character in upwards and downwards directions



Example of bit placement in symbol characters when direction of placement changes



Example of bit placement adjacent to Alignment Pattern



Symbol character arrangement in version 2-M symbol

Step 4 Masking Pattern selection
Apply the masking patterns defined in section 8.8.1 from standard [2] in turn and evaluate the results in accordance with 8.8.2. The Masking Pattern selected is referenced 011.

ISSN 2067 – 4074

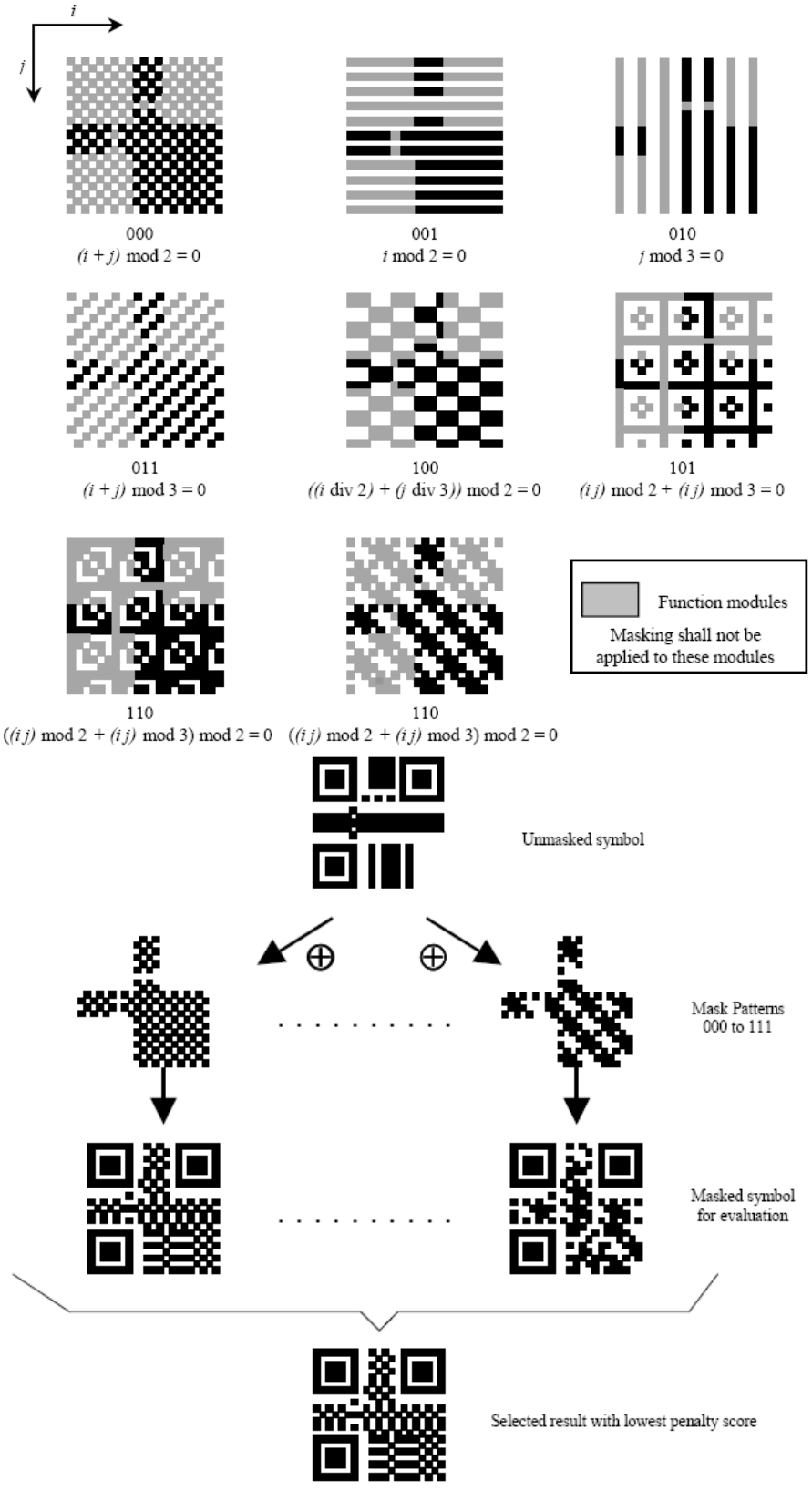


Figure 2.4. Masking Simmulation in Model 2

After performing the masking operation with each Mask Pattern in turn, the results shall be evaluated by scoring penalty points for each occurrence of the following features. The higher the

number of points, is the less acceptable the result.

Step 5: Format Information

Assume Error Correction Level M: **00**
and Mask Pattern Reference: **101**
Results Data: **00101**

The Format Information consists of a 15 bit sequence comprising 5 data bits and 10 BCH error correction bits. The Annex C in [2] describes the calculation of the error correction bits and the error correction decoding process. The BCH - Bose-Chaudhuri-Hocquenghem (15,5) code shall be used for error correction. The polynomial whose coefficient is the data bit string shall be divided by the generator polynomial $G(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$.
The coefficient string of the remainder polynomial shall be appended to the data bit string to form the (15,5) BCH code string. Finally, masking shall be applied by XOR-ing the bit string with **101010000010010** to ensure that the format information bit pattern is not all zeroes for any combination of Mask Pattern and Error Correction Level and to enable Model 2 symbols to be autodiscriminated from Model 1 symbols.

The BCH error correction calculation gives **1101011001** as the bit sequence to be added to the data, giving: **000111101011001** as the unmasked Format Information. XOR this bit stream with the mask **101010000010010**:
000111101011001 (raw bit stream)
101010000010010 (mask)
101101101001011 (Format Information to be placed in symbol – 15 bits)

For details please see table 25 and figure 19 in standard [2]:

Table 25 — Error correction level indicators

Error Correction Level	Binary indicator
L	01
M	00
Q	11
H	10

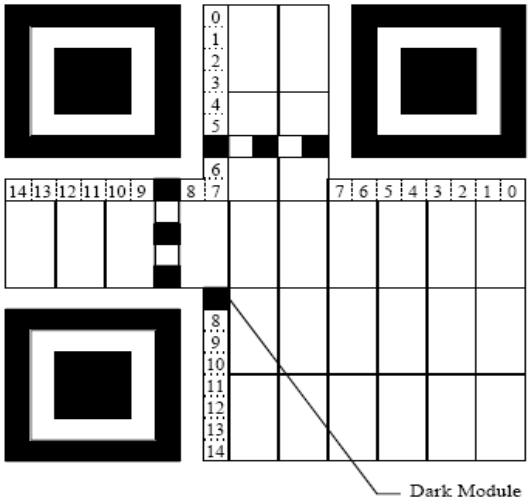


Figure 19 — Format Information positioning

Step 6: Final symbol construction
Apply the selected masking pattern to the encoding region of the symbol as described in section 8.8 from standard [2], and add Format Information modules in positions reserved in step 3.

Final version 1-M symbol encoding 01234567

ISSN 2067 – 4074

Before presenting the utility of barcode in the ticket of the public transportation system there is a business card encoded in QR code – created using tool from [9]:

Fig. 2.3 Business Card with QR code [9]

Table 2.2 shows the encoded information in QR code:

Table 2.2 QR Decoding of Business Card

***MECARD*:N:**Toma,Cristian;**ADR:**Calea Dorobantilor Ave., No. 15-17,Bucharest,7000,Romania;**TEL:**+40 21 319 19 00-310;**EMAIL:**cristian.toma@ie.ase.ro;**NOTE:**IT&C Security Master;**URL:**http://www.ism.ase.ro;

The following section presents the MMS and DRM topics which are the foundation for the ticket issuing procedure.

3. MMS using DRM

For complete reference regarding MMS = Multimedia Message Service, is recommended, Open Mobile Alliance W@P Forum [10] and especially [11] – MMS Architecture and [14] – MMS Encapsulation protocol. For complete OMA DRM – Digital Rights Management standards [15], [16] is recommended for study [17] – OMA DRM Architecture, [18] – OMA DRM Specs, [19] – OMA DRM Content format and author paper and book which contain topics about mobile digital rights management from [20] and [21].

The figure 3.1 presents a MMS without DRM – it encapsulates in this case:

- MMS standard headers
- SMIL headers
- Text file
- Image JPG file

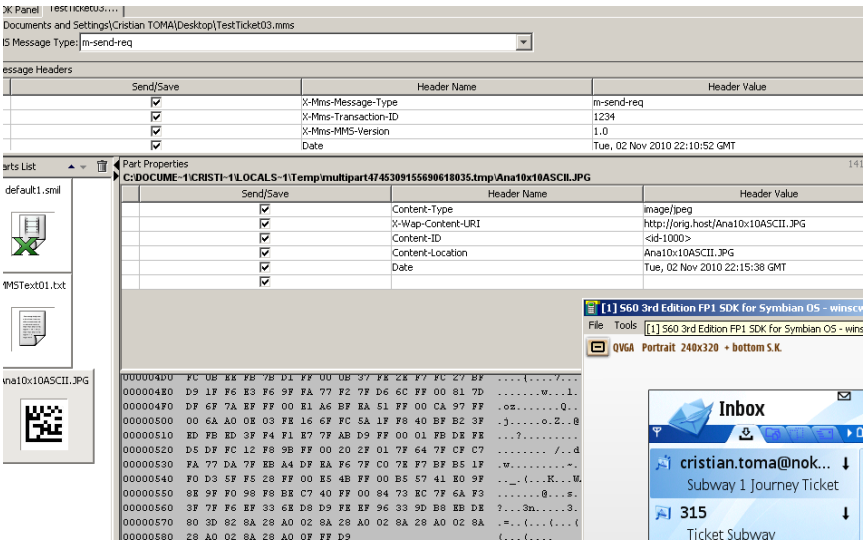


Figure 3.1 Ticket Test MMS without DRM

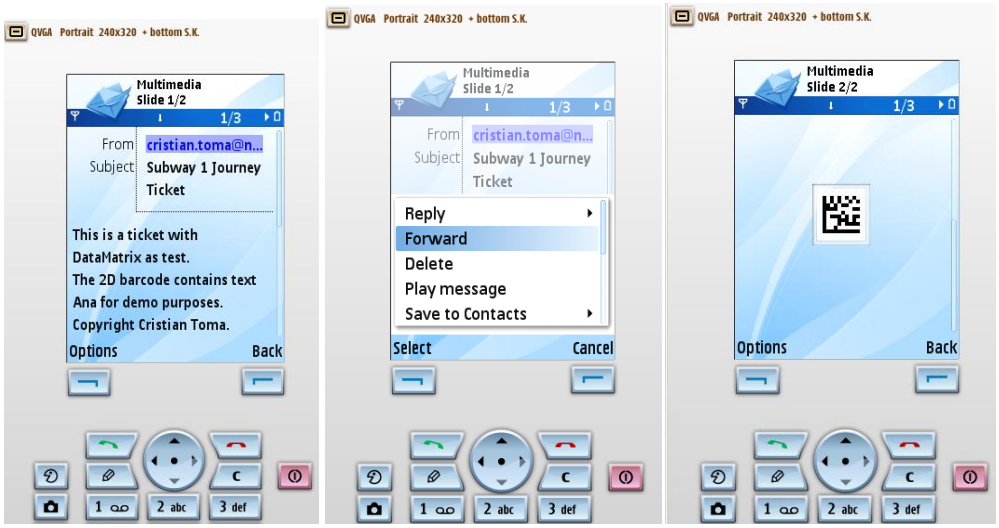


Figure 3.2 Ticket Test MMS Content without DRM

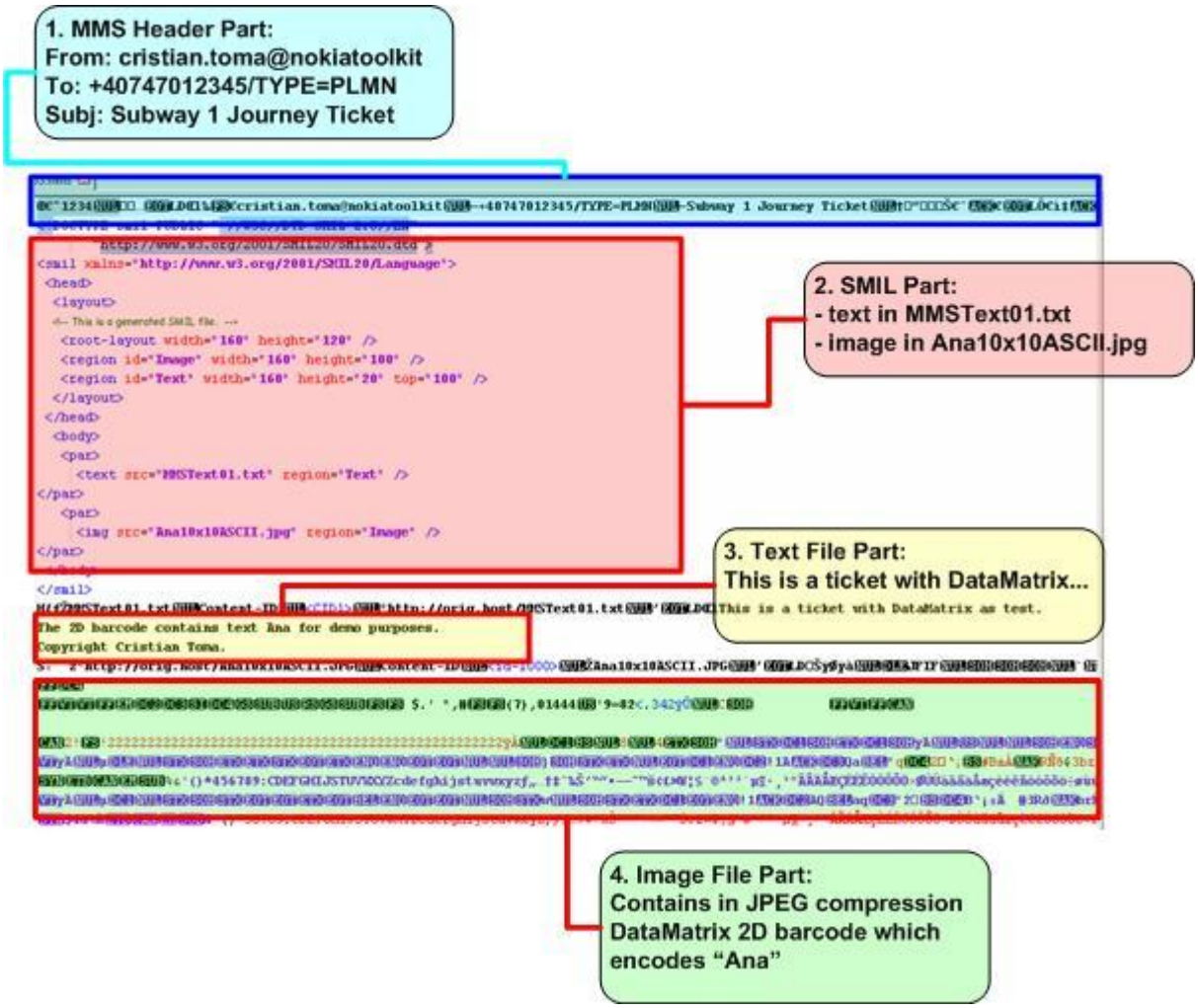
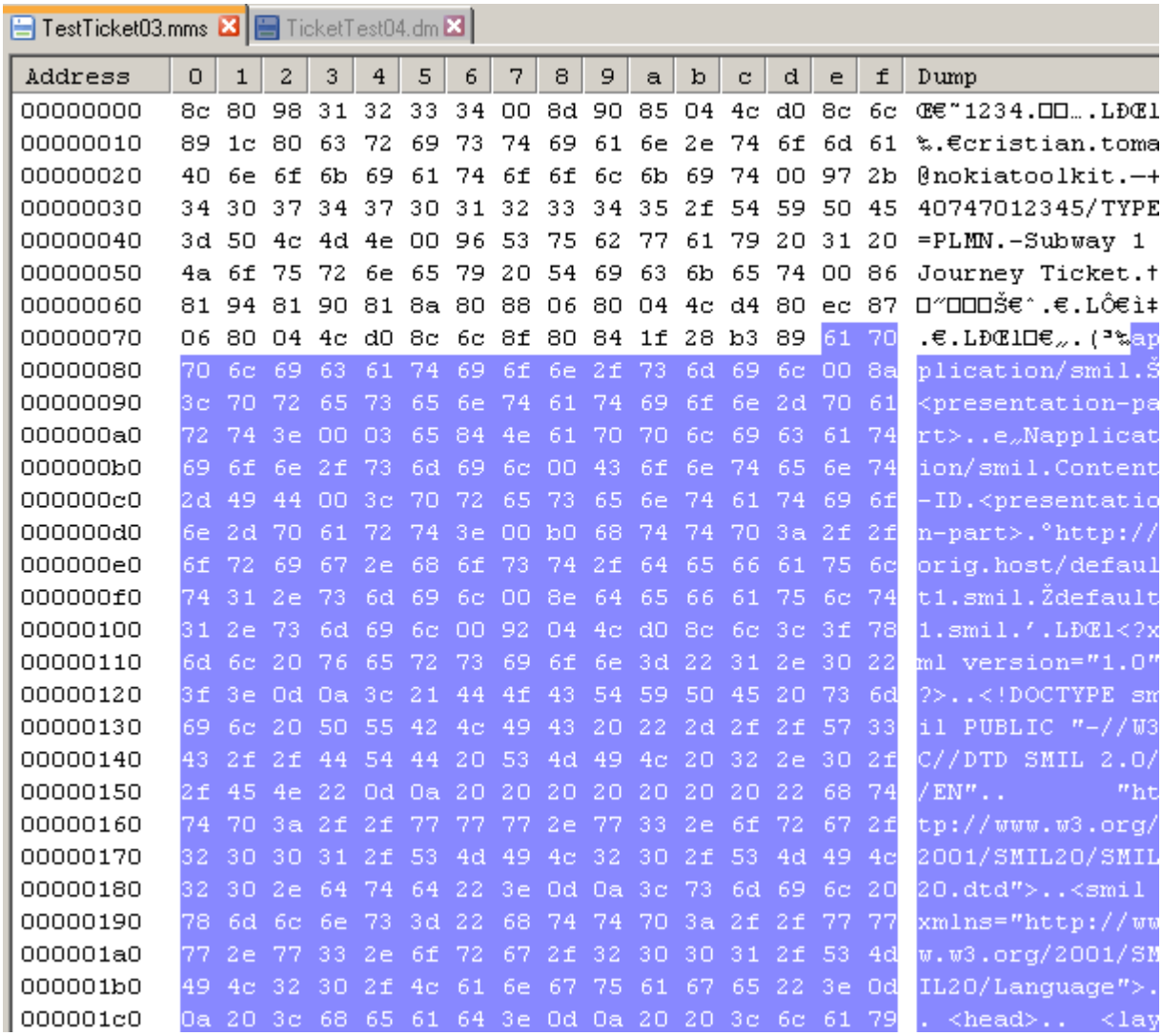


Figure 3.3 MMS parts format

ISSN 2067 – 4074


TestTicket03.mms | TicketTest04.dm

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	8c	80	98	31	32	33	34	00	8d	90	85	04	4c	d0	8c	6c	Œ€˜1234.□□….LÐŒl
00000010	89	1c	80	63	72	69	73	74	69	61	6e	2e	74	6f	6d	61	%.€cristian.toma
00000020	40	6e	6f	6b	69	61	74	6f	6f	6c	6b	69	74	00	97	2b	@nokiatoolkit.—+
00000030	34	30	37	34	37	30	31	32	33	34	35	2f	54	59	50	45	40747012345/TYPE
00000040	3d	50	4c	4d	4e	00	96	53	75	62	77	61	79	20	31	20	=PLMN.–Subway 1
00000050	4a	6f	75	72	6e	65	79	20	54	69	63	6b	65	74	00	86	Journey Ticket.†
00000060	81	94	81	90	81	8a	80	88	06	80	04	4c	d4	80	ec	87	□″□□□Šۀˆ.€.LÔ€ì‡
00000070	06	80	04	4c	d0	8c	6c	8f	80	84	1f	28	b3	89	61	70	.€.LÐŒl□€„.(³‰ap
00000080	70	6c	69	63	61	74	69	6f	6e	2f	73	6d	69	6c	00	8a	plication/smil.Š
00000090	3c	70	72	65	73	65	6e	74	61	74	69	6f	6e	2d	70	61	<presentation-pa
000000a0	72	74	3e	00	03	65	84	4e	61	70	70	6c	69	63	61	74	rt>..e„Napplicat
000000b0	69	6f	6e	2f	73	6d	69	6c	00	43	6f	6e	74	65	6e	74	ion/smil.Content
000000c0	2d	49	44	00	3c	70	72	65	73	65	6e	74	61	74	69	6f	-ID.<presentatio
000000d0	6e	2d	70	61	72	74	3e	00	b0	68	74	74	70	3a	2f	2f	n-part>.°http://
000000e0	6f	72	69	67	2e	68	6f	73	74	2f	64	65	66	61	75	6c	orig.host/defaul
000000f0	74	31	2e	73	6d	69	6c	00	8e	64	65	66	61	75	6c	74	t1.smil.Ždefault
00000100	31	2e	73	6d	69	6c	00	92	04	4c	d0	8c	6c	3c	3f	78	1.smil.'.LÐŒl<?x
00000110	6d	6c	20	76	65	72	73	69	6f	6e	3d	22	31	2e	30	22	ml version="1.0"
00000120	3f	3e	0d	0a	3c	21	44	4f	43	54	59	50	45	20	73	6d	?>..<!DOCTYPE sm
00000130	69	6c	20	50	55	42	4c	49	43	20	22	2d	2f	2f	57	33	il PUBLIC "-//W3
00000140	43	2f	2f	44	54	44	20	53	4d	49	4c	20	32	2e	30	2f	C//DTD SMIL 2.0/
00000150	2f	45	4e	22	0d	0a	20	20	20	20	20	20	20	20	22	68	/EN".. "h
00000160	74	70	3a	2f	2f	77	77	77	2e	77	33	2e	6f	72	67	2f	tp://www.w3.org/
00000170	32	30	30	31	2f	53	4d	49	4c	32	30	2f	53	4d	49	4c	2001/SMIL20/SMIL
00000180	32	30	2e	64	74	64	22	3e	0d	0a	3c	73	6d	69	6c	20	20.dtd">..<smil
00000190	78	6d	6c	6e	73	3d	22	68	74	74	70	3a	2f	2f	77	77	xmlns="http://ww
000001a0	77	2e	77	33	2e	6f	72	67	2f	32	30	30	31	2f	53	4d	w.w3.org/2001/SM
000001b0	49	4c	32	30	2f	4c	61	6e	67	75	61	67	65	22	3e	0d	IL20/Language">.
000001c0	0a	20	3c	68	65	61	64	3e	0d	0a	20	20	3c	6c	61	79	. <head>.. <lay

Figure 3.4. MMS binary format

Figure 3.2 shows the behavior of MMS encapsulated in figure 3.1. Because the MMS is not DRM prtected the "Forward" feature is activated at the mobile device.

The figure 3.3 shows the main parts of the MMS and 3.4 is the binary form representation in hex:

Table 3 – Binary representation of MMS
8C 84 .... Message type - MMS Message of type: *m-retrieve-conf:* A message received by an MMS device containing MMS media content. For *m-send-req:* A message sent from an MMS device containing MMS media content should be the value: **0x8C 0x80**
....
8D 90 .... MMS Version
85 04 4C D0 8C 6C – time and date in TLV ASN.1 DER format with values in secs from 1970 => aprox 357982 hours => 14915 days => 40.86 years => year 2010 sometimes in november
89 1c 80 63 72 69 73 74 69 61 6e 2e 74 6f 6d 61 40 6e 6f 6b 69 61 74 6f 6f 6c 6b 69 74 – 0x1c bytes length of 'From' field – 0x89 with value: cristian.toma@nokiatoolkit
00 – the fields separator
97 2b 34 30 37 34 37 30 31 32 33 34 35 2f 54 59 50 45 3d 50 4c 4d 4e – tag 0x97 is field 'To' with the value: +40747012345/TYPE=PLMN
00 – the fields separator
96 53 75 62 77 61 79 20 31 20 4a 6f 75 72 6e 65 79 20 54 69 63 6b 65 74

– tag 0x96 is field 'Subject' with value: Subway 1 Journey Ticket
00 – the fields separator
…
SMIL Part - Synchronized Multimedia Integration Language to control the presentation of the parts of MMS message.
…
TEXT Part – from file 'MMSText01.txt' with content: "This is a ticket with DataMatrix as test. The 2D barcode contains text Ana for demo purposes. Copyright Cristian Toma."
…
IMAGE Part – binary JPEG encodetion of DataMatrix for word "Ana"

The proposed model for 2D barcodes distribution is to generate OMA DRM MMS – minimum with "forward-lock" for each issued ticket – see figure 3.5 for different MMS delivery in terms of DRM.

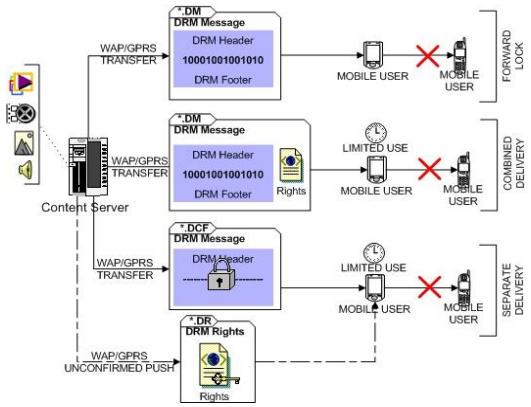


Figure 3.5. DRM Models for MMS delivery [20]

According to the OMA DRM standards and represented in figure 3.5, there are three DRM models for the content delivery:

- *Forward-lock* – the content provider sends to the mobile browser a binary file (image, movie, game or application) with special header and footer like in table 1 with "*dm*" extension. The mobile browser launches an application called the DRM agent that allows the browser to display and play the m-content without a "*Send*" option, so the end-user has no possibility of forwarding the content to another device via Bluetooth or MMS.

Table 3.1. Forward-lock Representation of "dm" file

--boundary-1 **Content-type: image/jpeg** **Content-Transfer-Encoding: binary** ÿØÿà…Binary representation of the M-CONTENT **--boundary-1--**

- *Combined-delivery* – before the binary content there is an XML representation of the "rights object" like in table 3.2 (encapsulated also in a "dm" file), which allows the browser to play only 3 times between 01.10.2010 – 01.12.2010 and does not allow it to forward the m-content.

Table 3.2. Combined Delivery Representation of "dm" file

--boundary-1 **Content-type: application/vnd.oma.drm.rights+xml** **Content-Transfer-Encoding: binary** *<o-ex:rights* *xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"* *xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"* *xmlns:ds="http://www.w3.org/2000/09/xmldsig#/">* *<o-ex:context>* *<o-dd:version>1.0</o-dd:version>* *</o-ex:context>* *<o-ex:agreement>* *<o-ex:asset> <o-ex:context>* *<o-dd:uid>cid:http://content-id-here</o-dd:uid>* *</o-ex:context></o-ex:asset>* *<o-ex:permission>* *<o-dd:play>* *<o-ex:constraint>* *<o-dd:count>3</o-dd:count>* *<o-dd:datetime>* *<o-dd:start>2010-10-01T20:59:10</o-dd:start>* *<o-dd:end>2010-12-*

ISSN 2067 – 4074

01T20:59:10</o-dd:end>
</o-dd:datetime>
</o-ex:constraint>
</o-dd:play>
</o-ex:permission>
</o-ex:agreement>
</o-ex:rights>

--boundary-1
Content-type: image/jpeg
Content-ID: <http://content-id-here>
Content-Transfer-Encoding: binary

ÿØÿà…Binary representation of the M-CONTENT
--boundary-1--

- *Separate-delivery* – the model allows the content provider to send the m-content that is encrypted with a symmetric key as in table 3.3 and 3.4. Therefore, within the separate delivery model, the content provider first sends the binary encrypted data with a header, encapsulated as in table 3.3 and figure 3.5 in a "dcf" file. The browser of the mobile device requests or receives the "rights object" file (the XML encapsulated in a "dr" file) from the URL included in "Rights-Issuer" field from "dcf" file. The rights object, if not request, can be pushed using WAP (Wireless Application Protocol) MMS – Multimedia Message Service or Push message (SI – Service Indicator or SL – Service Locator) mechanisms.

Table 3.3. Separated Delivery Representation of "dcf" file

image/jpegcid:http://content-id-here□gŽŒEncryption-Method: AES128CBC
Content-Name: "NameOfContent"
Rights-Issuer: http://rights-issuer.com/content
Content-Description: "DescriptionOfContent"
Content-Vendor: "VendorName"
Icon-Uri: http://vendor.com/content-icon.gif
"¶{…Binary encrypt representation of the M-CONTENT using AES-Rijndael symmetric key algorithm in CBC mode

Table 3.4. Separated Delivery Representation of "dr" file

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE o-ex:rights PUBLIC "-//OMA//DTD DRMREL 1.0//EN"
"http://www.oma.org/dtd/dr">
<o-ex:rights
xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"

xmlns:ds="http://www.w3.org/2000/09/xmldsig#/">
<o-ex:context>
<o-dd:version>1.0</o-dd:version>
</o-ex:context>
<o-ex:agreement>
<o-ex:asset>
<o-ex:context>
<o-dd:uid>cid:http://content-id-here</o-dd:uid>
</o-ex:context>
<ds:KeyInfo>
<ds:KeyValue>

joVbFmkmi3bSO6gC98HE1Q==
</ds:KeyValue>
</ds:KeyInfo>
</o-ex:asset>
<o-ex:permission>
<o-dd:play>
<o-ex:constraint>
<o-dd:count>2</o-dd:count>
<o-dd:datetime>
<o-dd:start>2006-09-27T20:59:10</o-dd:start>
<o-dd:end>2007-09-27T20:59:10</o-dd:end>
</o-dd:datetime>
</o-ex:constraint>
</o-dd:play>
</o-ex:permission>
</o-ex:agreement>
</o-ex:rights>

In conclusion, there are two ways of delivering the content rights object to the user, taking into consideration the

number of files that are sent to the mobile device:

- to the consuming devices, together with media object (DRM Forward Lock and Combined Delivery Model);
- sending the rights separately from media content (DRM Separate Delivery Model).

Regardless of which of the three models is implemented a *download descriptor file* such as in table 3.5 can be used in order to improve the user experience.

Table 3.5–Download Descriptor Representation "dd" file

```
<media
xmlns="http://www.openmobileallianc
e.org/xmlns/dd">
   <DDVersion>1.0</DDVersion>
   <name>Name Of Product</name>
   <size>1234</size>
   <type>image/jpg</type>
   <vendor>Media Vendor
Company</vendor>

<description>Description</description
>
   <objectURI>http://object-
url</objectURI>
   <iconURI>http://icon-
url</iconURI>
   <infoURL>http://info-url</infoURL>
   <nextURL>http://next-
url</nextURL>
   <installNotifyURI>
     http://install-notify-url
  </installNotifyURI>
  <installParam>-param1 -
param2</installParam>
</media>
```

The mobile device downloads the download descriptor file and the browser is redirected to the URL (the address between "<**objectURI**>" tag from "dd" file – table 5) that contains or generates the "dm" or "dcf" file depending on which of the DRM models present. The table 3.6 presents the MIME (Multipurpose Internet Mail Extensions) media types of the objects, according to the DRM message format.

Table 3.6. MIME media types

DRM method	**MIME media types**
Forward-lock	application/vnd.oma.drm.message
Combined delivery	application/vnd.oma.drm.message application/vnd.oma.drm.rights+xml
Separate delivery	application/vnd.oma.drm.rights+xml application/vnd.oma.drm.rights+wbxml application/vnd.oma.drm.content

The DRM message is based on a MIME multipart composite type in which one ore more objects are combined in a single body. The body of the DRM message must be according to the body of the multipart media type defined in RFC 2045 and 2046. The Digital Right Management message must contain one or two body parts, one for each object. If HTTP (Hyper Text Transfer Protocol) or a MIME compliant protocol is used to transport the Digital Right Management message, the boundary delimiter must be included as a parameter within the media type definition.

ISSN 2067 – 4074

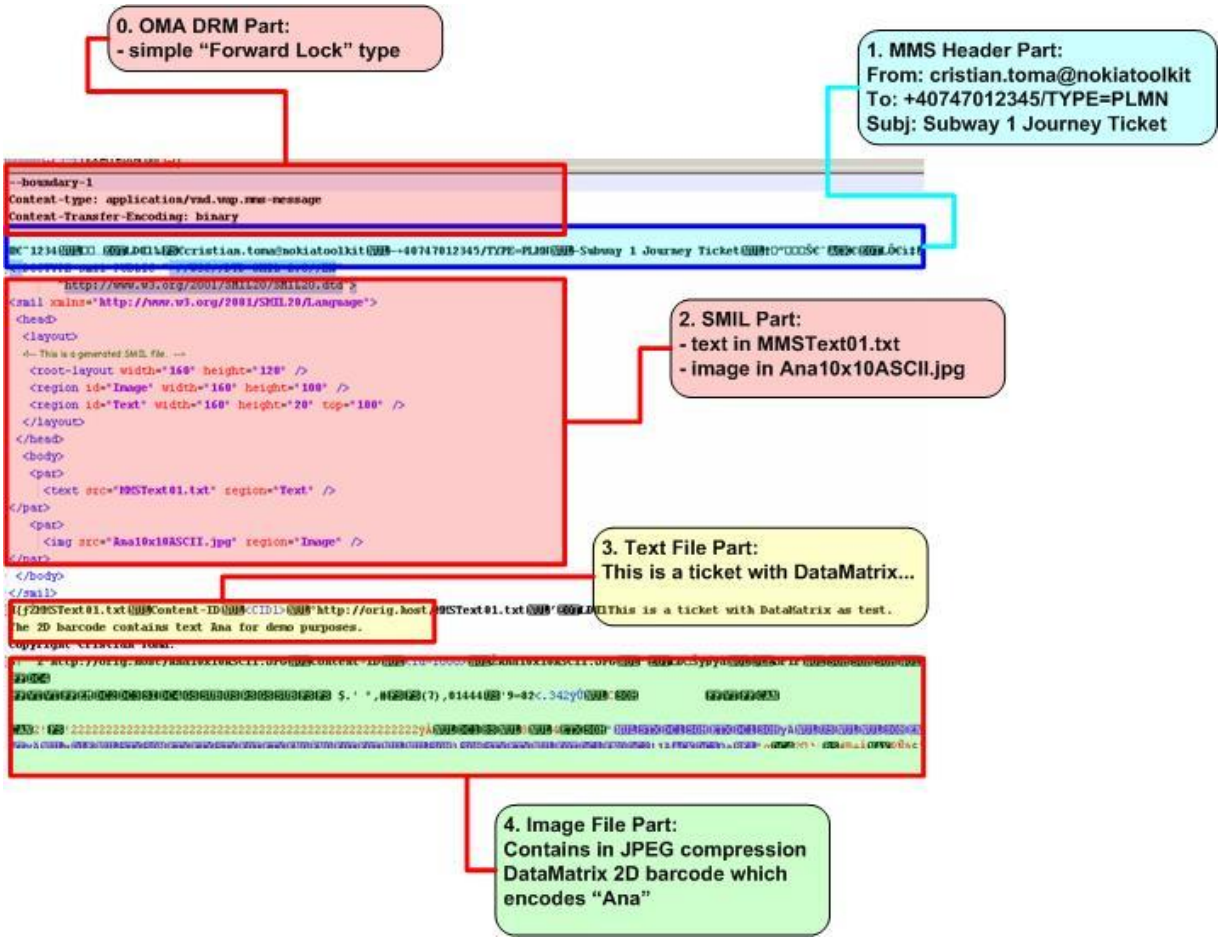


Figure 3.6. MMS with DRM main Parts

TestTicket03.mms | TicketTest04.dm

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	2d	2d	62	6f	75	6e	64	61	72	79	2d	31	0d	0a	43	6f	--boundary-1..Co
00000010	6e	74	65	6e	74	2d	74	79	70	65	3a	20	61	70	70	6c	ntent-type: appl
00000020	69	63	61	74	69	6f	6e	2f	76	6e	64	2e	77	61	70	2e	ication/vnd.wap.
00000030	6d	6d	73	2d	6d	65	73	73	61	67	65	0d	0a	43	6f	6e	mms-message..Con
00000040	74	65	6e	74	2d	54	72	61	6e	73	66	65	72	2d	45	6e	tent-Transfer-En
00000050	63	6f	64	69	6e	67	3a	20	62	69	6e	61	72	79	0d	0a	coding: binary..
00000060	0d	0a	8c	80	98	31	32	33	34	00	8d	90	85	04	4c	d0	..Œ€˜1234.□□….LÐ
00000070	8c	6c	89	1c	80	63	72	69	73	74	69	61	6e	2e	74	6f	Œl‰.€cristian.to
00000080	6d	61	40	6e	6f	6b	69	61	74	6f	6f	6c	6b	69	74	00	ma@nokiatoolkit.
00000090	97	2b	34	30	37	34	37	30	31	32	33	34	35	2f	54	59	—+40747012345/TY
000000a0	50	45	3d	50	4c	4d	4e	00	96	53	75	62	77	61	79	20	PE=PLMN.–Subway
000000b0	31	20	4a	6f	75	72	6e	65	79	20	54	69	63	6b	65	74	1 Journey Ticket
000000c0	00	86	81	94	81	90	81	8a	80	88	06	80	04	4c	d4	80	.†□″□□□Š€ˆ.€.LÔ€
000000d0	ec	87	06	80	04	4c	d0	8c	6c	8f	80	84	1f	28	b3	89	ì‡.€.LÐŒl□€„.(³‰
000000e0	61	70	70	6c	69	63	61	74	69	6f	6e	2f	73	6d	69	6c	application/smil
000000f0	00	8a	3c	70	72	65	73	65	6e	74	61	74	69	6f	6e	2d	.Š<presentation-
00000100	70	61	72	74	3e	00	03	65	84	4e	61	70	70	6c	69	63	part>..e„Napplic
00000110	61	74	69	6f	6e	2f	73	6d	69	6c	00	43	6f	6e	74	65	ation/smil.Conte
00000120	6e	74	2d	49	44	00	3c	70	72	65	73	65	6e	74	61	74	nt-ID.<presentat

Figure 3.7 MMS with DRM binary format

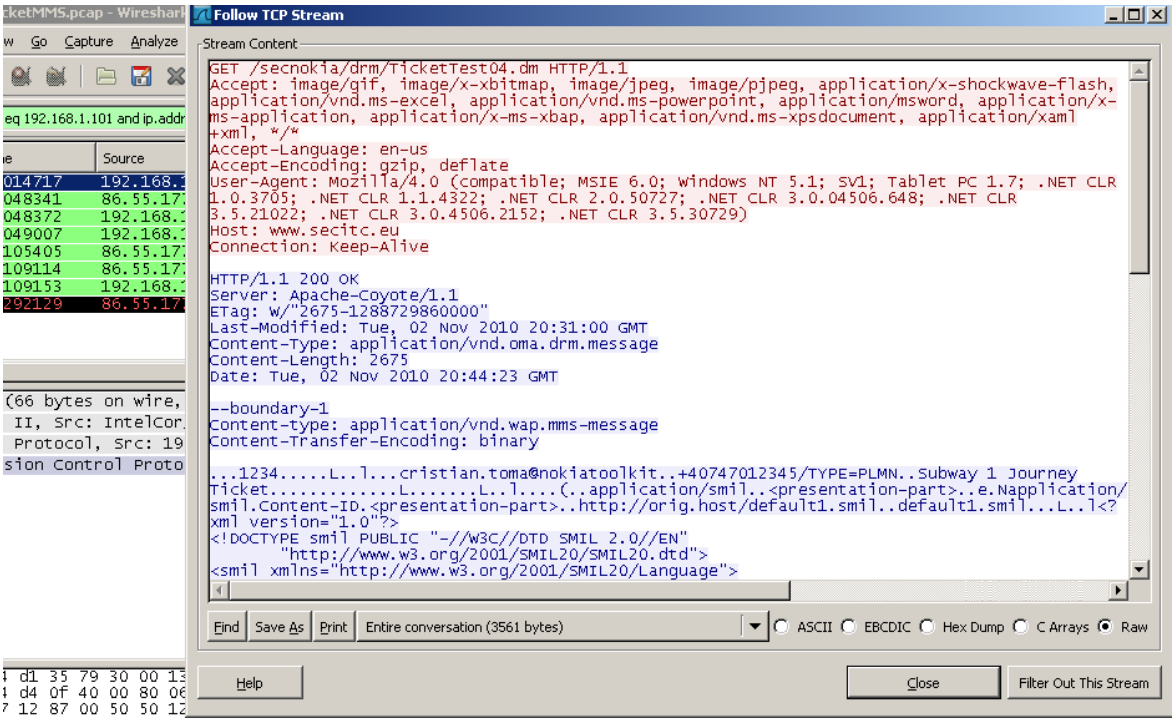


Figure 3.8. MMS with DRM HTTP analysis

Figure 3.6 is figure 3.3 but because of OMA DRM Forward Lock, it is an extra with text:
Table 3.7 OMA Header for Forward Lock
--boundary-1
Content-type: application/vnd.wap.mms-message
Content-Transfer-Encoding: binary
Figure 3.7 is figure 3.4 with the same binary interpretation but with the binary bytes as plus for the text from table 3.7

4. The Subway 2D Barcode Automatic Ticketing System – 2D BATS

The speed and the accuracy of reading process is a MUST in Public Transportation Ticketing Systems. Figure 4.1 presents the evolution of QR code – copyright [22]:

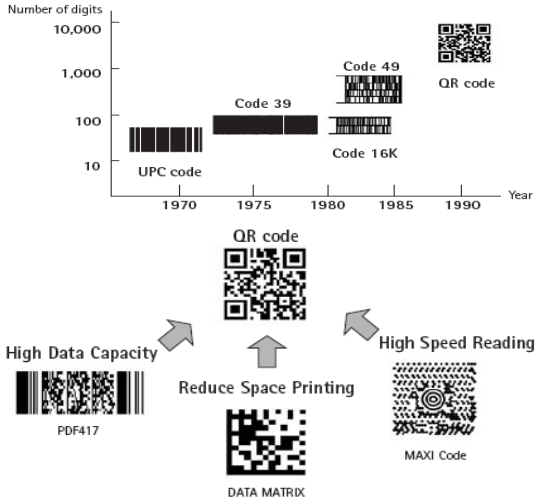


Figure 4.1. QR Code Evolution [22]

In 2D BATS will be used QR code for encoding tickets. The ticket representation is inspired from figure 2.3 and table 2.2, so, the purposed version of ticket is in table 4.1, 4.2 and figure 4.2:

ISSN 2067 – 4074

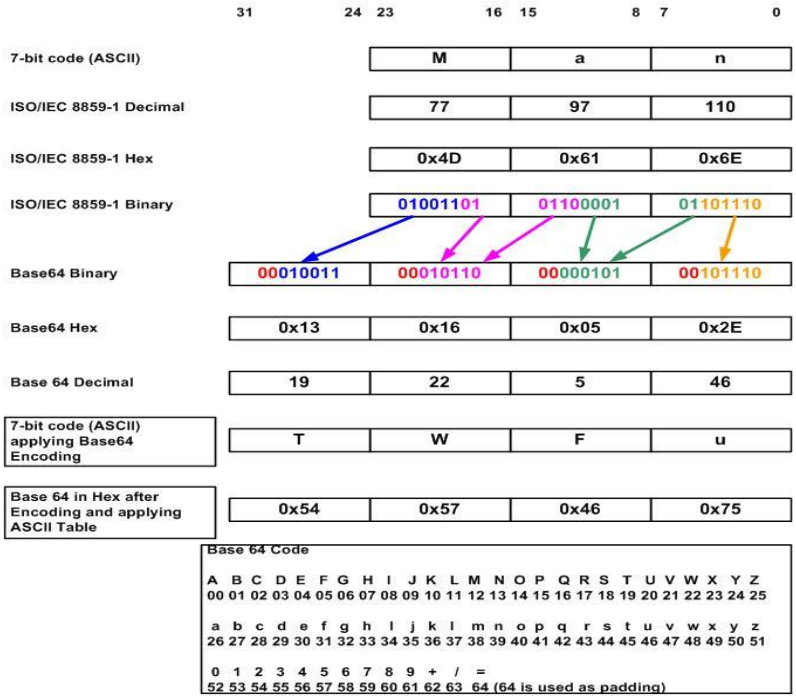


Figure 4.3. Base64 for word "Man"

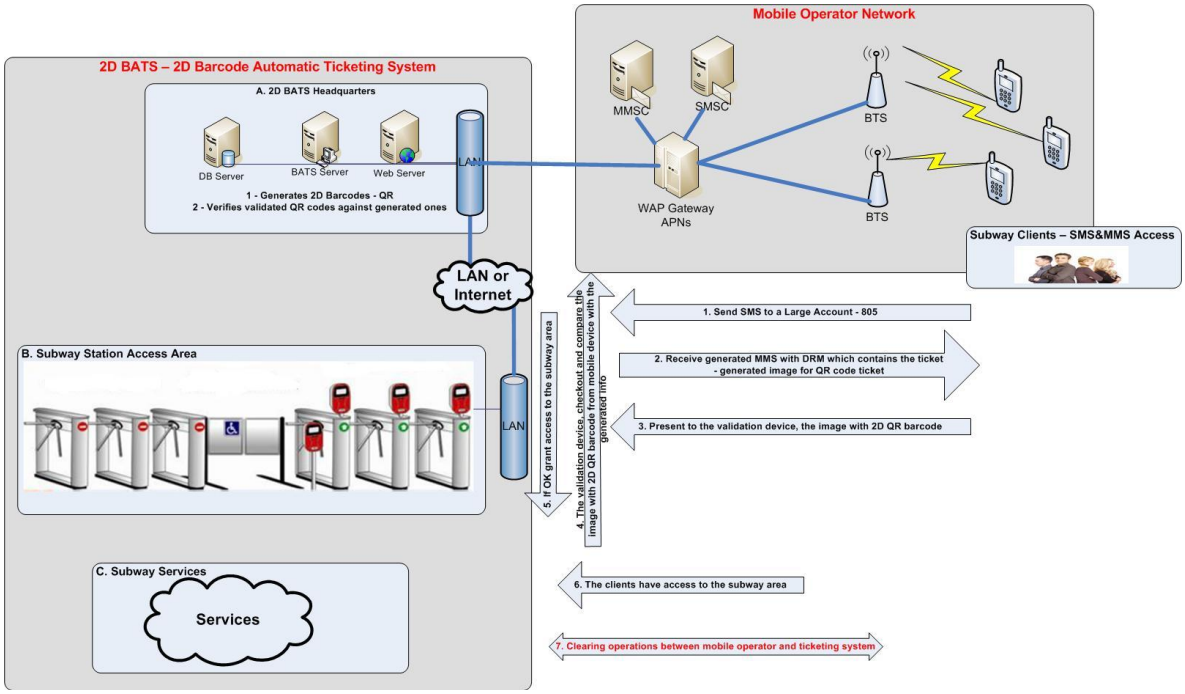


Figure 4.4 2D BATS Architeture

The data flow is the following:

- The clients access via SMS a large account with premium tax fee;
- The mobile operator sends the requests to the 2D BATS back-end systems;
- The back-end systems generate the 2D barcode QR encapsulated in a MMS with OMA DRM – minimum "forward lock";
- The MMS with DRM arrives to the clients via MMSC and the proper

APNs equipments – mobile operator premises;
- The MMS is received and the clients can not send the MMS content to them – first security level;
- The clients presents the MMS to the validation devices from the subway stations;
- The validation devices decode QR code, decrypt the content of the field ***ENCINFO***, and decode from Base64 in byte arrays.
- The validation devices prefrm security checks about info encapsulated in QR code - ticket and send info to the back-end servers of 2D BATS.
- If 2D BATS back-en systems confirm that everything is OK, the access in the station is granted by the validation device to the client.

5. Conclusion

The main advantages of 2D BATS solution are: the optimization, low cost and speed of the distribution chain which is in charge for the tickets issuing.

In the future, the optimization of the barcode info, the encryption/the signing process of the sensitive info from the tickets, the DRM improvement and the frontend/backend validation prcess enhancement, should be research topics for security issues in such kind of systems.

Parts of this research have been published in the Proceedings of the 3rd International Conference on Security for Information Technology and Communications, SECITC 2010 Conference (printed version).

References

[1] DataMatrix 2D barcode standard ISO/IEC 16022:2006

[2] QR 2D barcode standard ISO/IEC 18004:2000

[3] Harry E. Burke, *Automating Management Information Systems: Barcode Engineering and Implementation*, Thomson Learning Publishing House, ISBN 0-442-20712-3.

[4] Roger C. Palmer, *The Bar Code Book*, Helmers Publishing, ISBN 0-911261-09-5

[5] http://makebarcode.com/info/info.html

[6] http://en.wikipedia.org/wiki/Barcode

[7] http://www.asciitable.com

[8]http://www.qrme.co.uk/qr-code-resources/understanding-a-qr-code.html

[9]http://www.tec-it.com/online-demos/Business-Cards/Free-Business-Cards.aspx?lang=en

[10] http://www.openmobilealliance.org/Technical/wapindex.aspx

[11] http://www.openmobilealliance.org/tech/affiliates/wap/wap-205-mmsarchoverview-20010425-a.pdf

[12] http://www.openmobilealliance.org/tech/affiliates/wap/wap-206-mmsctr-20020115-a.pdf

[13] http://www.openmobilealliance.org/tech/affiliates/wap/wap-206_101-mmsctr-20011009-a.pdf

[14] http://www.openmobilealliance.org/tech/affiliates/wap/wap-209-mmsencapsulation-20020105-a.pdf

[15] http://www.openmobilealliance.org/Technical/release_program/drm_v1_0.aspx

[16] http://www.openmobilealliance.org/Technical/release_program/drm_v2_0.aspx

[17] http://www.openmobilealliance.org/Technical/release_program/docs/DRM/V2_0_2-20080723-A/OMA-AD-DRM-V2_0_1-20080226-A.pdf

[18] http://www.openmobilealliance.org/Technical/release_program/docs/DRM/V2_0_

ISSN 2067 – 4074

2-20080723-A/OMA-TS-DRM_DRM-V2_0_2-20080723-A.pdf

[19] http://www.openmobilealliance.org/Technical/release_program/docs/DRM/V2_0_2-20080723-A/OMA-TS-DRM_DCF-V2_0_2-20080723-A.pdf

[20] Cristian TOMA, *Security in Software Distributed Platforms*, ASE Publishing House, Bucharest, 2008, ISBN 978-606-505-125-6

[21] Ion IVAN, Cristian TOMA, Catalin BOJA, Marius POPA - Secure architectures for the Digital Rights Management of the M-Content, WSEAS Transactions on Computers, Issue 3, Volume 6, November 2006, ISSN 1109 – 2750 (http://www.wseas.org)

[22] Tan Jin Soon, *QR Code*, Automatic Data Capture Technical Committee Presentation