Enhancing Cost and Security of Arabic SMS Messages over Mobile Phone Network

Abdullah A. Abdullah
College of Computer Sciences and Mathematics
University of Mosul, Iraq

Received on: 27/08/2008 **Accepted on: 23/11/2008**

ABSTRACT

This paper investigates a novel algorithm for compressing and encrypting Arabic short text messages (SMS messages). Short text messages are used in cellular networks. Compression is required for saving the transmission energy or to use bandwidth in an efficient manner in addition to save the user money while the end-to-end effective encryption is required for security providence. This work succeeded to overcome small size limitation of the SMS message by changing Arabic characters coding from Unicode to Base64 coding scheme and developing a runt version of lossless Huffman coding scheme. Examples are shown where the application of the text compressor for short message services offering more than three times the capacity compared to a standard message.

Keywords: SMS compression, Huffman coding, Transposition key**.**

تحسين تكلفة وأمن الرسائل النصية العربية القصيرة عبر شبكة الهاتف المحمول

عبد الله عبد الله

كلية علوم الحاسوب والرياضيات، جامعة الموصل

تاريخ الاستلام: 2008/08/27 **تاريخ القبول: 2008/11/23**

الملخص

في هذا البحث تم تقديم خوارزمية جديدة لكبس وتشفير الرسائل النصية العربية القصيرة، والتي تستخدم في شبكات الهاتف الخلوي. إن الفائدة الأساسية لعملية الكبس هي لتوفير طاقة النقل اللازمة أو لنقل البيانات بأسلوب كفوء بالإضافة إلى توفير المال للمستخدم، بينما تعتبر عملية التشفير أساسية لتوفير سرية مناسبة للرسالة. في هذا العمل تم تجاوز مشكلة صغر الحجم المتاح للرسالة النصية القصيرة وذلك بتغيير نمط تمثيل الحروف العربية من التمثيل الموحد (Unicode) إلى التمثيل (Base64)، بعدها تم تطوير نسخة مختزلة لطريقة كبس هوفمان تناسب هذا الحجم الصغير. لقد ظهر من خلال تطبيق الطريقة المقترحة لكبس الرسائل النصية الصغيرة أن الزيادة في حجم الرسالة وصلت لأكثر من ثلاثة أضعاف حجم الرسالة التقليدية.

الكلمات الرئيسية: كبس الرسائل النصية القصيرة، كبس هوفمان، مفتاح تبديل.

I. INTRODUCTION

1. Mobile and SMS messages

The mobile phone is the most successful new technologies of the past two decades [13]. In addition to making voice communication mobile, the mobile phone brought to light a new form of communication: SMS (Short Message Service) or text messaging. Some researchers even argue that it is SMS – rather than voice calls - that has been the major force in the adoption of mobile phones. SMS's popularity was due to the controlled cost that SMS provides, and the efficiency of its asynchronous communication model [16]. The recent exploration of research of SMS as a separate communication medium is found in [15], which focuses solely on SMS communication.

The short message service (SMS) was developed as part of the Global System for Mobile Communications (GSM 03.40) standard [5] and it allows mobile systems and other network-connected devices to exchange short text messages with a maximum length of 160 characters. The length limit is caused by the way that SMS is transmitted. It usually rides on the control channels, the same frequencies or time slots used for call setup information by mobile phones. This means that users can send or receive SMS messages while they are making a phone call, though they need a hands-free kit to read the screen or type on the keypad [2]. SMS was commercially introduced in 1992.

SMS is a well established technology which has widespread use around the globe, is quick, efficient and reliable. It can be used in many fields such as to access banking details, to access local information services like traffic announcements, weather forecasts, news broadcasting etc. It has also been used in television programs to vote for people to stay in the show or be removed [9] [7].

A simplified view of an SMS message traversing a GSM-based system from submission to delivery was found in [21], including Submitting, routing and wireless delivery for a message.

The following can be named as some of the advantages of SMS

1. Communication is possible when the network is busy.
2. We can exchange SMS messages while making telephone calls.
3. SMS messages can be sent in offline mode.

SMS has been called the 'killer' application of mobile phones, as its usage exceeded all expectations. Some reasons given for the huge growth include low cost, asynchronous nature (users can reflect before sending and reply at their leisure) and potential for private or quiet use [11].

2. SMS (Short Message Service) specifications

The SMS message, as specified by the Etsi organization[5] (documents GSM 03.40 and GSM 03.38), can be up to 160 characters long,

where each character is 7 bits according to the 7-bit default alphabet. Eight-bit messages (max 140 characters) are usually not viewable by the phones as text messages; instead they are used for data in e.g. smart messaging (images and ringing tones). 16-bit messages (max 70 characters) are used for Unicode (UCS2) text messages, viewable by most phones. A 16-bit text message of class 0 will on some phones appear as a Flash SMS (aka blinking SMS or alert SMS).

II. MOTIVATION

1. SMS Compression

Compression of short messages is a vital operation for low complexity entities such as mobile phones or wireless sensors. As is the case with any computing system, yet, particularly in embedded systems, data compression is one of the most important applications due to the restricted resources available [14][3]. In the mobile phone's world, the compression would allow users to increase the number of characters of their short message service (SMS). As it seems obvious to compress these messages, state of the art text compressor based on LZ77 and others would fail to compress such small messages ending up with more data than the original [6]. Therefore, a totally new concept is needed to compress short messages. The transmission of compressed data over cellular networks is done in a transparent way (so still not more than 160 characters or 140 bytes are transmitted in one SMS) and therefore a compressor/decompressor is needed on both ends, as illustrated in Figure 1. The compression will pay off if the energy spent by the compression and decompression is lower than the energy saved for the transmission, such that the compression and decompression is done only once, but the gain for shorter transmission time and in turn less power consumption is achieved by each hop.

Another advantage of the compression is that the time spent on the wireless medium is shorter and therefore more capacity is available [19]. This is especially important for information aggregation, where a central entity wakes up thousands of nodes asking them to provide some information. Therefore, compression of short messages to be conveyed over the wireless medium seems to be promising in terms of power, costs, and bandwidth savings. The complexity introduced in terms of computational power and memory usage will be investigated and reported to be low.

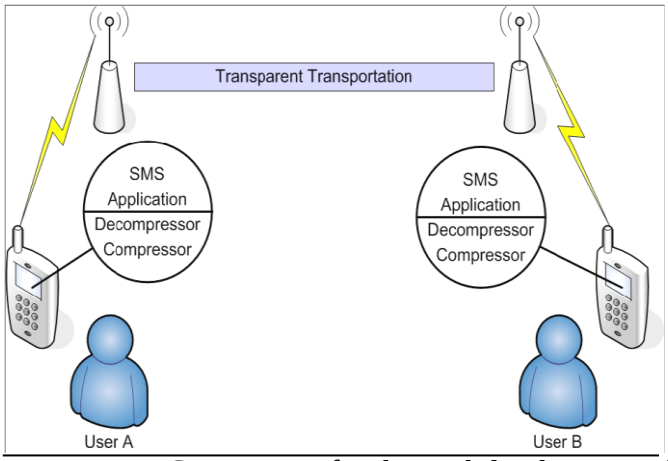


Figure. 1. Text Compressor for the Mobile Phone World.

2. SMS security

The contents of SMS messages are visible to the network operator's systems and personnel. Therefore, SMS is not an appropriate technology for secure communications. Most users do not realize how easy it is to intercept messages. It would likely be a relatively complex to hack into a telecom provider's systems to obtain the content of SMS messages, but finding staff privileged to look at SMS messages and persuading them to reveal the contents is much easier. Gartner Research has already expressed reservations about security in U.K. trials of SMS voting in local elections held in May 2002. Enterprises, including governments, cannot use SMS in its present state for any confidential communication. Enterprises seeking secure communication channels to mobile employees should consider encrypted end-to-end solutions on devices having additional security features. The underlying specifications and technology for SMS transmission leave many security gaps. These gaps make SMS vulnerable to [12][21]:

1. Snooping: On device, at the store and forward network elements
2. SMS Interception: Over the air, in wired network
3. Spoofing: Using commercial tools, own SMS gateway Modification
4. Using conventional hacking techniques

In this paper, we intended to overcome the SMS message size limitations in order to decrease transmission cost for user and network as well as adding suitable and effective end-to-end encryption method to improve its reliability.

III. RELATED WORK

In [4], an optimal statistical model is adaptively constructed from a short text message and transmitted to the decoder. Thus, such an approach is not useful for short message compression, as the overall compression ratio

would suffer from the additional size of the context model. The recent paper in [8] uses syllables for compression of short text files larger than 3 kBytes. A related work in the field of very short text files is the study in [6], where a tree machine is employed as a static context model. It is shown that failure occurs for short messages (compression starts for files larger than 1000 Bytes). In contrast to the work in [19], the model is organized as a tree and allocates 500 KBytes of memory, which makes the proposed method less feasible for a mobile device. In [18] the compression for the smaller data models was improved by using a modified hash function. Furthermore, a methodology for the design and analysis of low complexity data-models together with extended performance results are given in [17].

IV. PRACTICAL

The aim of this work is to add cost and security enhancements to the Arabic SMS messages, which can be send over GSM networks, these enhancements involve two branches:

1. Encoding (or compression) with suitable algorithm
2. Encryption method, which must be effective, simple and fast.

The overall algorithm we applied should be suitable to some mandatory constraints:

1. Mobile hardware capabilities constraints (low speed).
2. Mobile memory constraints.
3. The size of SMS messages which is considered very short (140 byte or 1120 bit).

Fixed-length code versus variable-length code

A fixed length code is based on the idea that, all the letters in a given alphabet have same probability of occurrence (i.e. equal frequency). ASCII is an example of a fixed length code. There are 100 printable characters in the ASCII character set, and a few non-printable characters, giving 128 total characters. Since lg 128 = 7, ASCII requires 7 bits to represent each character. The ASCII character set treats each character in the alphabet equally, and makes no assumptions about the frequency with which each character occurs.

A variable length code is based on the idea that for a given alphabet, some letters occur more frequently than others. This is the basis for much of information theory, and this fact is exploited in compression algorithms to use as few bits as possible to encode data without “losing” information. More sophisticated compression techniques can use compression techniques that actually discard information like image and video data. However, for text compression, we do not want to have characters discarded as part of the compression, so a text compression requires a unique decodability condition

of the compression algorithm [10][6]. Our enhancement includes both of them. Firstly in Fixed-length code, by minimizing the representation of Arabic alphabet codes from standard Unicode (16 bit per symbol) into a compact alphabet version (6 bit per symbol), and finally in variable-length code, by using a modified runt version of Huffman [1] coding scheme to suite SMS message small size.

Fixed-length code enhancement

Arabic text usually coded using Unicode coding [20] (16 bit per character), this make the Arabic SMS message can contain a maximum of 70 characters while default alphabet SMS message can contain up to 160 characters (7 bit per character).

@	Δ	PS	0	¡	P	¿	p
£	-	!	1	A	Q	a	q
$	Φ	"	2	B	R	b	r
¥	Γ	#	3	C	S	c	s
è	Λ	¤	4	D	T	d	t
é	Ω	%	5	E	U	e	u
ù	Π	&	6	F	V	f	v
ì	Ψ	'	7	G	W	g	w
ò	Σ	(	8	H	X	h	x
Ç	Θ	)	9	I	Y	i	y
FL	Ξ	*	:	J	Z	j	z
Ø	)1	+	;	K	Ä	k	ä
ø	Æ	,	<	L	Ö	l	ö
RC	Æ	-	=	M	Ñ	m	ñ
Å	ß	.	>	N	Ü	n	ü
å	É	/	?	O	§	o	à

Table. 1. illustrates the 7-bit default alphabet as specified by GSM 03.38

As shown in the Table 1, we can notice that, many symbols are not used in normal Arabic language (i.e. like Ç , Ψ ,ü , ¤, … etc). Since we can derive from Unicode a compact Arabic alphabet containing only 64 symbols (i.e. Base64 coding scheme), including standard Arabic characters and some

common used symbols (i.e. like +, - , * , / , (,) , : , … etc) together with their corresponding original Unicode, can be shown in Table 2.
Each symbol from our suggested Arabic alphabet now can be represented in (6 bit only) instead of (16 bit or 7 bit).This scheme extends the message size to be nearby 186 symbols per message (1120 bit / 6 bit).

Variable-length code enhancement

After changing the Arabic alphabet -coding scheme from the Unicode to Base64 coding scheme in order to reduce the alphabet size, then message will travel into the following three main steps:

1. Apply Run-length compression algorithm using an escape character ('@') when there is a benefit, in order to reduce the message length. this step added since repeated symbols chains are very common in real life user messages.
2. Apply our proposed runt frequency table Huffman algorithm, which we suggest to make Huffman coding applicable in SMS messages due to its very small length (160 characters) only.
3. Apply transpose encryption; this simple and fast encryption method will reduce the computation time as compared with other complex encryption methods. It will use a transpose key derived from the receiver phone number which is entered by the sender in order to strength it. The derived key will be used twice, firstly in rearranging the frequency table before encoding process started in order to scatter
4. their values, and secondly to perform XORing operation on the header part of the message only. This security procedure will be sufficient to provide a reliable encryption scheme, since without the header information there is no way to know which are the used characters, how many bits occupied by each character frequency value and what are these frequencies, which considered necessary information to any intruder to decode the received message.

Base64 code	Char	Unicode	Base64 code	Char	Unicode	Base64 code	Char	Unicode
0	٠	1632	22	ش	1588	44	ء	1569
1	١	1633	23	ص	1589	45	ة	1577
2	٢	1634	24	ض	1590	46	Cr	0010
3	٣	1635	25	ط	1591	47	lf	0013
4	٤	1636	26	ظ	1592	48	Sp	0032
5	٥	1637	27	ع	1593	49	!	0033
6	٦	1638	28	غ	1594	50	%	0037
7	٧	1639	29	ف	1601	51	(	0040
8	٨	1640	30	ق	1602	52	)	0041
9	٩	1641	31	ك	1603	53	*	0042
10	ا	1575	32	ل	1604	54	+	0043
11	ب	1576	33	م	1605	55	,	0044
12	ت	1578	34	ن	1606	56	-	0045
13	ث	1579	35	ه	1607	57	.	0046
14	ج	1580	36	و	1608	58	/	0047
15	ح	1581	37	ي	1610	59	:	0058
16	خ	1582	38	أ	1571	60	;	0059
17	د	1583	39	إ	1573	61	=	0061
18	ذ	1584	40	آ	1570	62	?	0063
19	ر	1585	41	ى	1609	63	@	0064
20	ز	1586	42	ؤ	1572			
21	س	1587	43	ئ	1574			

Table. 2. proposed Base64 Arabic alphabet

Proposed Sender Algorithm

The following abbreviations list are used throughout the algorithm:
ArabicSMS: array to store the Arabic SMS message.
PN: sender Phone Number.
FreqList: Frequencies List
USV: Used Symbols Vector
FFS: Frequency Field Size
RFT: Runt Frequency Table.

1. Read the entered Arabic SMS message (in Unicode) call it ArabicSMS of size 70 letter as a full message (may be another size) then read the receiver PHONE NUMBER, call it PN. ArabicSMS[70]=

   "********مرحبا علي, سأهاتفك لاحقا هذا المساء, اتمنى ان تكون بخير********"

   Receiver phone number: PN = 1740405
   Message size = 70 symbol in 1120 bit
2. Applying Run Length compression using '@' as an escape symbol, in order to reduce redundancy characters chain like (" ******* "), which are very common in real life messages. This step will be applied only if it will minimize message size , in this case ArabicSMS[61] =

   "@*٧مرحبا علي, سأهاتفك لاحقا هذا المساء, اتمنى ان تكون بخير@*٨"

   Note: if the '@' character appeared repeated in original message like "@@@@@", then it will be replaced as "@@٥", also if the repeated character are more than 3 times then run length considered useful to be applied ,else there is no need.
   Message size = 61 symbol in 976 bit
3. Converting each Unicode character in ArabicSMS into corresponding code from our proposed 6-bit codes.
   Message size = 61 symbol in 366 bit

6-bit code	**63**	**53**	**07**	**33**	**19**	**15**	**11**	**10**	**48**	**27**	**32**	…
Symbol	**@**	*	٧	**م**	**ر**	**ح**	**ب**	**ا**	**sp**	**ع**	**ل**	**…**
Unicode	**0064**	**0042**	**1639**	**1605**	**1585**	**1581**	**1576**	**1575**	**0032**	**1593**	**1604**	**…**

4. Calculate ArabicSMS symbols frequencies and store it in frequency table FreqList[n] where n < m (n = 27 used characters, m = 61)
5. Sort FreqList in descending order by frequencies values as in Table 3, this table will be stored later in a runt (or compact) manner as a part of message header data, so that the receiver can extract it then build the code-word binary tree (Huffman tree) which then used to encode message characters into variable length codes.

6bit code	char	Frq	6bit code	char	Frq
10	ا	**9**	**35**	ه	**2**
48		**9**	**31**	ك	**2**
33	م	**3**	**07**	٧	**1**
32	ل	**3**	**27**	ع	**1**
12	ت	**3**	**40**	أ	**1**
34	ن	**3**	**29**	ف	**1**
63	@	**2**	**30**	ق	**1**
53	*	**2**	**18**	ذ	**1**
19	ر	**2**	**44**	ء	**1**
15	ح	**2**	**41**	ى	**1**
11	ب	**2**	**36**	و	**1**
37	ي	**2**	**16**	خ	**1**
55	,	**2**	**08**	٨	**1**
21	س	**2**			

Table. 3. Sorted frequencies

6. Calculate Used-Symbols-Vector (USV), which is 64 bit (8 bytes) stream corresponding the 64 alphabet characters, for example if bit 15th value was '1' this indicate that character 'ح' was used in this message, else considered absent. USV in hexadecimal will be 80 9D 2D E8 3F 13 A1 80.

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
15-00	1	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0
31-16	1	1	1	0	1	0	0	0	0	0	1	0	1	1	0	1
47-32	0	0	0	1	0	0	1	1	0	0	1	1	1	1	1	1
63-48	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

7. Compute bits needed to represent each coded symbol frequency,
   Frequency-Field-Size (FFS) = $\log_2$ (max (FreqList[n])) bits.
   ex: FFS = $\log_2$ (9) = 4 bits to represent each frequency.
   Then frequencies can be stored in (27 * 4) bits only, which we call Runt Frequency Table RFT.
8. Apply transpose encryption to frequency table before encoding process started in order to scatter their values through the following steps:

- Shift with rotate the phone number,
  ex: PN = "1740405" $\rightarrow$ PN = "5174040"

This step is necessary to overcome a situation when PN is originally ordered like "1346889", in this case the key will be "1234567" which is not effective.

- Order PN in ascending order, the index is the derived Key.

shifted	5	1	7	4	0	4	0
Ordered	0	0	1	4	4	5	7
index	4	6	1	3	5	0	2

Key = 4613502 in hexadecimal is 04 61 35 02

Then derived key will be used in rearranging the frequency table before encoding process started in order to scatter their values. Tailer characters (if less than 7) remain unchanged for simplicity. After that, we must replace each character in the message with the corresponding one according to Key, so the message now rearranged depending on the key sequence.

char	ا		م	ل	ت	ن	@	*	ر	ح	ب	ي	,	س	…
Key	**4**	**6**	**1**	**3**	**5**	**0**	**2**	**4**	**6**	**1**	**3**	**5**	**0**	**2**	**…**
New char	ت	@		ل	ن	ا	م	ي	س	ر	ب	,	*	ح	….

The message will be:

"سمي٨,يخ@نقوا@تا@تن اى@تل حتء *@أذت@لتر ٧ت @حفأتنهق@*,عل@ميك سربت"

9. Build Huffman binary tree from frequencies to produce new code words for only used symbols. The algorithm works by constructing a binary tree from the bottom up, using the frequency counts of the symbols to repeatedly merge sub-trees together. Intuitively, the symbols that are more frequent should occur higher in the tree and the symbols that are less frequent should be lower in the tree.
   Conceptually, the algorithm creates a weighted node for each symbol, and repeatedly merges the lowest-frequency nodes into the tree, adding the weights cumulatively as in Figure 2.
10. Encode each symbol in ArabicSMS by corresponding code word.
    The first 10 characters of original message coded by new code words are shown below; the difference in storage size is obvious, that 20-byte (10*16 bit) message size can be stored now in (46 bit) only.
    Now previous message with 122-byte (61*16 bit) size can coded into 267 bit only which result from:
    Coded-message data size = $\sum$ FreqList[i] * CWS[i], I = 1...n
    Where CWS[i] equal to Code Word Size in bits of i-th character.

م	ي	ك		س	ر	ب	ت	@	ع	…
00001	**00101**	**11111**	**1101**	**10101**	**01001**	**11001**	**000**	**100**	**010001**	…

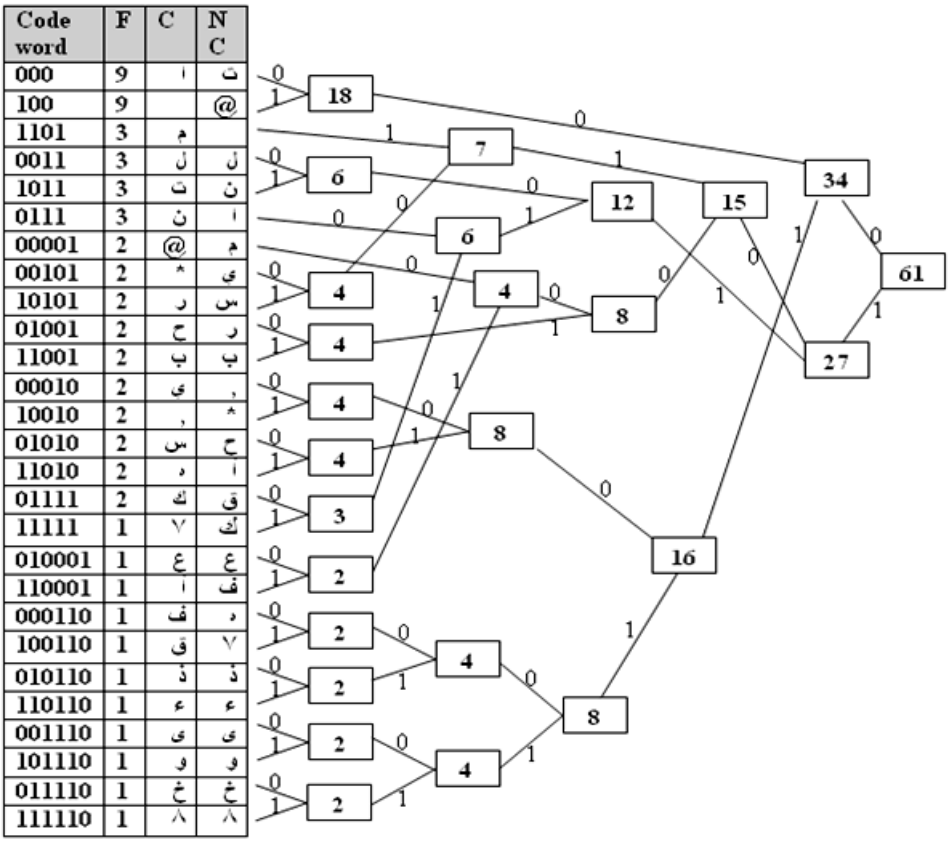


Figure.2. resulted Huffman coding binary tree

11. Store Header data which consist of three parts as in Table 4:
    - USV (size = 64 bits).
    - FFS value (size = 6 bits).
    - RFT (size = (FFS * n) bits).

    Such that each frequency value of FreqList resides in FFS bits, this chain of frequencies should ordered by the original alphabet-indexing scheme.

    Overall message size (MS) calculated as:

    MS = USV + FFS + RFT + Coded-data stream

    In this example:

    MS = 64 + 6 + 108 + 267 = 445 bits, instead of 1120 bits of original message.

USV 64 bit	FFS 6 bit	RFT 108 bit (4 *27)											Coded Data
		٧	٨	ا	ب	ت	ح	ذ	ر	س	ع	…	…
9D 2D E8 3F 13 A1 80	04	1	1	9	2	3	2	1	2	2	1	…	…

Table. 4. New message structure

12. Appling a transpose encryption, such that the transpose key which was derived from the receiver phone number PN and previously used in rearranging the frequency table now will be XORed with the header fields USV and FFS (i.e. encrypt Header only).

	Message Header (178 bit)		Coded Data …
	USV + FFS	RFT	
Bit stream	80 9D 2D E8 3F 13 …	…	
XORing	⊕ ⊕ ⊕ ⊕ ⊕ ⊕	…	
Repeated key	02 35 61 04 02 35 …	…	
Final message	82 A8 48 E8 39 26 …	…	

13. Send new coded SMS message to the receiver with PN.

Proposed Receiver Algorithm

1. Read the received coded Arabic SMS message, with receiver phone number PN.
2. Derive the transposition key from PN, then XOR it with message header fields USV and FFS only.
3. Extract Header data fields, which consist of USV, FFS and RFT.
4. Determine what are the used symbols from USV field.
5. Extract frequency values from RFT using FFS value.
6. Sort frequencies in descending order by frequencies value.
7. Apply transpose encryption to return the original frequency table.
8. Build Huffman binary tree from frequencies list.
9. Decode each symbol in the received message by the corresponding base64 code.
10. Converting each base64 code back to its original Unicode representation.
11. Applying Run Length decompression using '@' as an escape symbol, in order to get the original message.

V. Experimental results and Compression ratio

Naturally, one of the most important metrics of efficiency for compression algorithms is compression ratio. Compression ratio can be defined as the ratio of size of the original text to that of the coded text,

defined as (Size of original text) /(Size of coded text), For above example, Compression ratio = 1120 / 445 = 2.516

In Figure 3, we draw the compression ratio for several different messages varied in size and number of used characters. We found that as the message size increases the compression ratio also increases, which is expected because when the message size increases the frequency of the symbols will increase, so we expect to have better compression on large messages.

Additionally, results show that the size of proposed header for the message that depends on number of used characters does not significantly affect compression ratio.

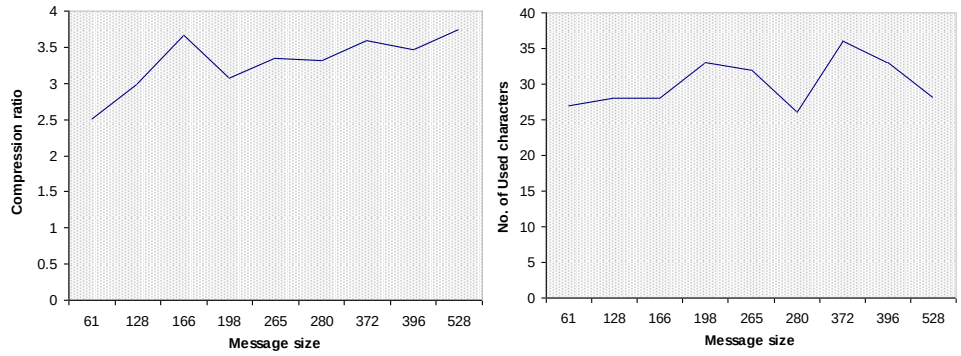
As a result, from any compression process, a storage space is freed. However, since Huffman coding depends on variable length codes, we cannot determine exactly how many characters can resides into the freed space. Therefore, we will divide the freed space by the Average code-word length in order to estimate the count of characters, which can be added to the message bounded by original message size (1120 bits). To illustrate this we calculate:

Gained bits = 1120 – 445 = 675 bits

Average code-word length = 267 / 61 = 4.377 bits

Expected additional characters = 675 / 4.377= 154 characters.

Estimated new message size = 70 + 154 = 224 character



a) Compression ratio b) No. of used characters

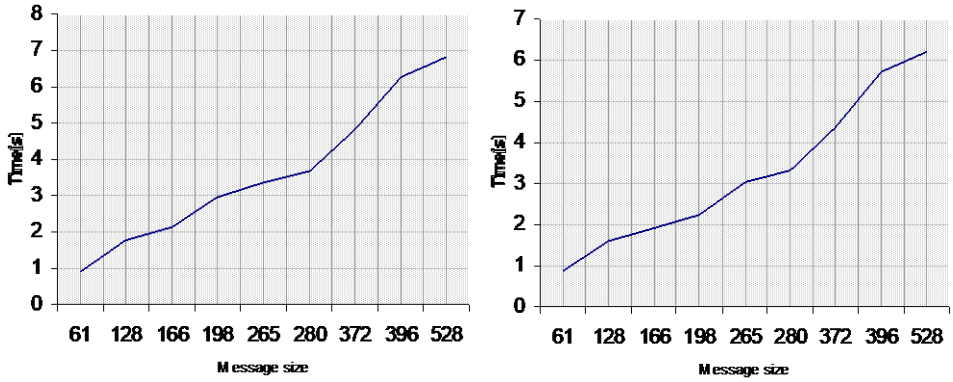
Figure. 3. Experimental results

We can notice that, if the user adds often characters of code word bits less than 4.377 bits (like 'ا','م','ن' … etc), we can get more than 154 additional characters, that is mean the compression ratio will increase. while if the added characters were often have code word bits more than 4.377 bits (like 'ز','ح','ف' … etc), then we can get less than 154 additional characters, that is mean the compression ratio will decrease.

In this example, we reach to 2.51 % compression ratio; this rate is considered a high ratio especially when compared with the small data size constraint.

Another important aspect in analyzing results is the compression time, here we present the time needed for a stand alone compression on the Nokia 6300 as example with its 238MHz processor. In Figure 4 the time needed to compress or decompress is given respectively.

In general the time needed for compression is obviously more than the once needed for decompression that is due to compression extra steps such as (calculating frequencies). We can also notice that message length is considered main factor in increasing time. Therefore, for a message length of 128, the Nokia 6300 needs 1.77 seconds for compression and 1.61 seconds for decompression. While a message length of 528 needs 6.83 seconds and 6.21 seconds for compression and decompression respectively.



a) Compression time b) Decompression time

Figure. 4. Compression and Decompression time in seconds

VI. CONCLUSION

This paper investigates a novel algorithm for compressing and encrypting Arabic short text messages. The algorithm can be applied in cellular mobile communication systems. It has been found that the compression of Arabic SMS saves space and reduces transmission time. Also it is found that as the message size increases the compression ratio also increases, that is expected because when the message size increases the frequency of the symbols will increase, so there is a great expectation to have better compression on large messages. Additionally, results show that the size of proposed header for the message does not significantly affect compression ratio. However, Arabic users can send nearby 3 in 1 encrypted SMS messages.

REFERENCES

[1] A. Huffman, 1952, "A Method for the Construction of Minimum Redundancy Codes," Proc. IRE, Vol. 40, pp. 1098-1101.

[2] A. Dornan, "The Essential Guide to Wireless Communications Applications", published by Prentice Hall PTR, ISBN 0-13-031716-0.

[3] C. Sadler and M. Martonosi, 2006, "Data compression algorithms for energy-constrained devices in delay tolerant networks". In Sen-Sys'06, pages 265.278.

[4] E. Hatton, 1995, "Samc–efficient semi-adaptive data compression," in Proceedings of the IBM 1995 conference of the Centre for Advanced Studies on Collaborative research, Toronto, Ontario, Canada, p. 29.

[5] European Telecommunications Standards Institute (ETSI), GSM 03.40 v7.4.0, Digital cellular telecommunications system (Phase 2+), Technical realization of the Short Message Service (SMS), ETSI 2000, http://www.etsi.org.

[6] G. Korodi, J. Rissanen, and I. Tabus, 2005, "Lossless data compression using optimal tree machines," in Proceedings of the IEEE Data Compression Conference (DCC'05), pp. 348–357.

[7] G. Le Bodic, "Mobile Messaging Technologies and Services – SMS, EMS & MMS", published by John Wiley & Sons Ltd, ISBN 0-470-84876-6.

[8] J. Lansky and M. Zemlicka, 2006, "Compression of small text files using syllables," in Proceedings of the IEEE Data Compression Conference (DCC'06), p. 458.

[9] Latest Mobile, GSM, Global, Handset, Base Station, & Regional Cellular Statistics http://www.cellular.co.za/stats/stats-main.htm.

[10] M. Nelson and J.-L. Gailly, 1995. "The Data Compression Book", 2nd Edition. M&T Brooks, San Mateo, CA.

[11] M. Shirali-Shahreza, 2006, "Stealth Steganography in SMS," Proceedings of the third IEEE and IFIP Int. Conference on Wireless and Optical Communications Networks (WOCN 2006), Bangalore, India.

[12] Network Security Solutions, 2006. White paper on SMS vulnerabilities and XMS technology enabling mCOMMERCE. http://www.mynetsec.com February.

[13] K., Nyiri, 2003. "Mobile Communication: Essays on Cognition and Community". Published by Passagen Verlag, Vienna.

[14] P. Havinga and G. Smit, 2000. "Design techniques for low-power systems". Journal of Systems Architecture: the EUROMICRO Journal archive, 46(1):1.21.

[15] R., Harper, L. Palen, and A.S. Taylor, 2005, "The Inside Text". Springer, Germany.

[16] S. Jenson, 2005, Default Thinking: Why consumer products fail. In The Inside Text. Eds. Harper, R., Palen, L. and Taylor A. Springer.

[17] S. Rein and C. G¨uhmann, "A free library for context modeling with hash-functions– part i," Wavelet Application Group, Tech. Rep., May 2005. [Online]. Available: http://www.mdt.tu-berlin.de.

[18] S. Rein, C. G¨uhmann, and F. Fitzek, 2006. "Low complexity compression of short messages," in Proceedings of the IEEE Data Compression Conference (DCC'06), pp. 123–132.

[19] S. Rein, C. G¨uhmann, and F. Fitzek, 2006. "Compression of Short Text on Embedded Systems". J. Computers, Vol. 1, No. 6, September.

[20] Unicode Standard, Version 5.0, 1991-2008 (ISBN 0-321-48091-0), [Online] at http://www.unicode.org/versions/Unicode5.0.0 .

[21] W. Enck, P. Traynor, P. McDaniel, and T. La Porta, 2005, "Exploiting Open Functionality in SMS Capable Cellular Networks", In Proceedings of ACM CCS'05, November 7–11, Alexandria, Virginia, USA.