Arithmetic Coding Modification to Compress SMS

Ario Yudo Husodo[#1], Rinaldi Munir[*2]

[#*]*Informatics Department, School of Electrical Engineering and Informatics, Institut Teknologi Bandung*
Jalan Ganesha 10 Bandung

if17017@students.if.itb.ac.id
rinaldi@informatika.org

***Abstract*— This paper proposes an effective method to compress SMS by doing some modifications to arithmetic coding data compression mechanism. The aim of this proposal is to optimize the maximum character capacity of SMS body. Every character in SMS is mostly encoded in 7 bit and the maximum capacity of one SMS is only 1120 bit. Those SMS characteristics require a very efficient compression method to compress SMS.**

Arithmetic coding is a compression mechanism that works by converting a data message to a real code number between 0 and 1. Arithmetic coding provides nearly optimal data compression. However, it requires additional memory space in compressed-data to save arithmetic coding probability table for decompressing the compressed-data. Besides, it requires high-precision and effective encoder-decoder to calculate and represent its code number (compressed-data).

In very limited data space like SMS, the need of additional memory space to save arithmetic coding probability table is inefficient. It will make the compressed-SMS size bigger than the original SMS (uncompressed SMS) size. To overcome this inefficiency, in this paper, the need of memory space is erased. This paper proposes semi dynamic probability table usage to compress and decompress SMS for overcoming the inefficient need of memory space. To more optimize the effectiveness and efficiency of proposed-method compression ratio, this paper also proposes a smart data representation to represent code number so that the number of bits needed to represent compressed-SMS can be well-minimized. By using this smart data representation, 2k digit decimal code number value in base-10 can be written by only using k default GSM 7 bit characters.

The proposed compression mechanism in this paper has been researched plainly in mobile phone that uses Android operating system. The SMS data test language used on the research is Bahasa Indonesia. Based on the research, the compression ratio of proposed compression mechanism is vary depends on the content of SMS. The average compression ratio of proposed compression mechanism is 71%, while the maximum compression ratio is able to reach less than 25%, i.e. 500 character SMS can be compressed to 121 character SMS.

***Keywords* — arithmetic coding, modification, SMS, compression.**

I. INTRODUCTION

A. Arithmetic Coding

Arithmetic coding is a compression mechanism that works by converting a data message to a real code number between 0 and 1 [1]. To compress a data, arithmetic coding requires a probability table of characters contained in the data. Probability table is a table containing probability range of existing characters in a data which is built based on the existing characters frequency in the data itself. The smaller the range to generate code number, the higher bit number is needed to represent the code number [2]. Arithmetic coding only needs usual arithmetic operation to compress and decompress a message. On arithmetic coding, compression encoding is not done to every single character but it is done straight to the message itself. Basically, arithmetic coding is able to compress a message with compressed-message result near to the message entropy value.

There are two main weaknesses of arithmetic coding. The first one is that it needs memory space in compressed-data to save its range probability table for decompressing the compressed-data. If arithmetic coding is used to compressed small-size data, the existence of the range probability table in compressed-data will make the compressed-data bigger in size than the original data itself. The second weakness of arithmetic coding is that it requires encoder-decoder with high-precision value. Encoder is a machine to compress a message; meanwhile decoder is a machine to decompress a compressed-message. If encoder or decoder doesn't have ability to calculate long mantissa with precision value, the decompressed message of compressed-message can be different from the original message.

B. Short Message Service (SMS)

Short Message Service (SMS) is a bi-directional communication service to send a text message via wireless communication system. The common SMS uses default GSM 7 bit alphabet encoding system. The maximum capacity of a SMS is 1120 bits [3]. It means that it is only able to contain maximum 160 standard characters encoded by default GSM 7 bit alphabet. Nowadays, there are three main encoding systems for SMS: the default GSM 7-bit alphabet, the 8-bit data alphabet, and the 16-bit UTF-16 alphabet. No matter the encoding system used to encode SMS characters, the maximum capacity of usual SMS is only 1120 bits.

There are two main parts of SMS, first is header, and second is body. SMS header consists of instruction sets related to components working on SMS service network, like receiver destination number and date information about the time when SMS is sent. In another hand, SMS body is a part of SMS containing the main message that SMS sender want to send. Because the part of SMS header must be readable by SMS provider and because the most SMS encoding system nowadays uses default GSM 7 bit alphabet, this paper will only continue discussion on an efficient method to compress

SMS body that uses default GSM 7 bit alphabet. To facilitate the easy call of proposed SMS compression mechanism in this paper, the proposed mechanism is named with ACHA, stand for Arithmetic Coding Hybrid Ario.

II. Proposed SMS Compression Mechanism

In making an efficient method to compress SMS body, two modification aspects of arithmetic coding are proposed in this paper.

A. Semi Dynamic Probability Table Usage

Every compression mechanism needs conversion table to compress and decompress a data. Usually, this conversion table built based on the data message itself. In pure arithmetic coding mechanism, the conversion table often called range probability table. Because the range probability table adapts to original data content, the range probability in arithmetic coding is also often called with dynamic range probability table. In arithmetic coding mechanism, to decompress a compressed-message correctly (match with original message), decoder needs the same range probability table used by encoder when encoder compressing the original message. To enable decoder having the same range probability table, usually the range probability used by arithmetic coding encoder is sent along with the compressed-data. Usually, the range probability is included in the beginning of compressed-data and it is called with compressed-data header.

For compressing a big-size data by using arithmetic coding mechanism, the compressed-data header doesn't give any significant difference of compressed-data size. But if the original data size is small, the existence of the compressed-data header can add very significant difference of compressed-data size. It could enlarge the compression ratio of a data message significantly. Compression ratio is a ratio between compressed-data size compared to original-data size. The smaller compression ratio of a compression mechanism, the better performance of the compression mechanism is.

$$\text{Compression Ratio} = \frac{Compressed-Data\ Size}{Original-Data\ Size}$$

Because SMS body only have maximum capacity of 1120 bit characters, SMS can be categorized as a small-size data message. It means that if SMS is compressed by using pure arithmetic coding mechanism, addition of compressed-data header is able to make the compressed-SMS size bigger than the original-SMS size. To overcome this problem, the existence of compressed-data header should be erased.

The most conventional way for erasing the existence of compressed-data header is by using static range probability table to compress any SMS body. It means that encoder and decoder use one-fit-all table to compress and decompress any kind of SMS body. This method is very simple but the efficiency of arithmetic coding mechanism will be decreased drastically. The decreased of arithmetic coding efficiency occurs because the one-fit-all table also contains range probability data of characters that do not exist in the original SMS body. This data definitely decrease the table range to generate code number, in another word, this data will increase the bit number needed to represent the code number.

To overcome the weaknesses of dynamic and static range probability table, this paper proposed-mechanism –ACHA– proposes a semi-dynamic range probability table for compressing and decompressing SMS. Semi dynamic range probability table is a dynamic range probability table that is built from some static range probability sub table. A static range probability sub table consists of static characters frequency that is predicted (based on probability and statistical calculation) will appear in original SMS. Different from conventional static range probability table, a static range probability sub table doesn't contain data frequency of all characters that possible to appear in original SMS, it only contains data frequency of some characters that possible to appear in original SMS. Data in a static range probability sub table is a part of data in usual static range probability table.

In ACHA mechanism, data in conventional static range probability table is divided to *n* ACHA static range probability sub tables (the number of *n* will be discussed later). Every single ACHA static range probability sub table contain almost-equal number of data. Any single data contained in ACHA static range probability sub table-*j* is never contained in ACHA static range probability sub table-*k*, where $j \neq k$. In ACHA mechanism, the main range probability table for compressing and decompressing SMS is built from the suitable ACHA static range probability sub tables.

In ACHA mechanism, after encoder reads the whole SMS body, encoder determines which ACHA sub tables (ACHA static range probability sub tables) should be used to build main range probability table. The configuration of ACHA sub tables used by encoder to compress original data is written only by using few characters (default GSM 7 bit character) – these characters are called **Conf_Char** (configuration character). After deciding the ACHA sub tables' configuration, ACHA build the main range probability table then encoder starts compression process just like common arithmetic-coding compression mechanism. In the end of compression process, encoder put the Conf_Char in the beginning of compressed-data as a part of the compressed-SMS body.

When compressed-SMS received by decoder, decoder read the Conf_Char, then decoder interprets the Conf_Char to build main range probability table to decode the whole compressed-SMS. After the main range probability table is built from ACHA sub tables based on Conf_Char, decoder decompresses the compressed-SMS just like common arithmetic-coding decompression mechanism.

If there are n ACHA sub tables, then call each ACHA sub tables T1, T2, .., and Tn. To build the main range probability table, each ACHA sub tables only has two possibilities: to be used as part of the main range probability table or not. Based on that fact, for n ACHA sub tables, there are 2^n possibilities of sub table configuration used on building the main range probability table. In ACHA mechanism, the configuration is represented by efficient Conf_Char (The representation process of Conf_Char will be discussed in section II.B).

To more understand the ACHA sub table mechanism, take a situation that we want to compress SMS in Bahasa Indonesia with message “ayah saya” (means “my father” in English). To simplify the calculation, assume we only have 16 recognized letters in SMS: {<space>,a,d,e,f,g,h,i,j,m,o,s,y,z,1,2} with conventional static data frequency of each characters is shown in Table 1. Table 1 shows the simulated average probability of each 16-character appearance number in 10,000 characters contained in SMS. Take the number of ACHA sub tables is 4. The four ACHA sub tables is Shown in Table 2, where each sub tables data is sorted by the characters frequency. Table 3 shows the overall main range probability used to compress message “ayah saya”.

TABLE I
SIMULATED AVERAGE PROBABILITY APPEARANCE NUMBER OF 16-RECOGNIZED CHARACTER IN 10,000 CHARACTERS IN SMS

Character	Frequency
<space>	1500
a	1400
s	1000
d	900
e	900
f	800
g	700
i	600
m	600
h	500
o	500
y	400
1	50
j	50
2	50
z	50

TABLE 2
ACHA SUB TABLES FOR TABLE 1

Sub Table T1

Character	Frequency
<space>	1500
a	1400
s	1000
d	900

Sub Table T2

Character	Frequency
e	900
f	800
g	700
i	600

Sub Table T3

Character	Frequency
m	600
h	500
o	500
y	400

Sub Table T4

Character	Frequency
1	50
j	50
2	50
z	50

TABLE 3
MAIN CHARACTER FREQUENCY TABLE FROM TABLE 2 TO COMPRESS MESSAGE “AYAH SAYA”

Character	Frequency
<space>	1500
a	1400
s	1000
d	900
m	600
h	500
o	500
y	400

From Table 3, to compress “ayah saya”, there are only two ACHA sub tables used to compress the message: T1 and T3. There is no character in T2 or T4 used in original message “ayah saya”, so T2 and T4 is not used to build the main character frequency table (Table 3). Compared with Table 1 where there are 16 data frequency saved, Table 3 only saves 8 data frequency. It means that the usage of ACHA sub tables eliminate significantly unused data frequency in main probability table for compressing or decompressing data. By implementing the ACHA semi-dynamic range probability table usage, the range to generate code number in ACHA mechanism will be bigger than the range in static range probability method, thus the bit number needed to represent the code number in ACHA will be more efficient than in static range probability method. Table 4 shows the overall main range probability table used to compress message “ayah saya”. After the main range probability table –Table 4– built, encoder starts compression process just like common arithmetic-coding compression mechanism.

TABLE 4
MAIN RANGE PROBABILITY TABLE FROM TABLE 3 TO COMPRESS MESSAGE “AYAH SAYA”

Character	Probability	Range	Low Range	High Range
<space>	1500/6800	0.00 – 0.2206	0.00	0.2206
a	1400/6800	0.2206 – 0.4265	0.2206	0.4265
s	1000/6800	0.4265 – 0.5735	0.4265	0.5735
d	900/6800	0.5735 – 0.7058	0.5735	0.7058
m	600/6800	0.7058 – 0.794	0.7058	0.794
h	500/6800	0.794 – 0.8675	0.794	0.8675
o	500/6800	0.8675 – 0.941	0.8675	0.941
y	400/6800	0.941 – 1.00	0.941	1.00

B. Smart Compressed-Data Representation

In arithmetic coding mechanism, a data is converted to a decimal code number. The longer the original message, the more mantissa needed to represent the compressed-message is. Primitive data structures such *float* or *double* practically is not good to be used as the code number type. The reason is because the both primitive data structures have limitation in number of mantissa they saves, besides, the mantissa precision they saved is not very precise.

The implementation target of ACHA mechanism is Android-operating-system-base mobile phone. Because Android use java programming language, to represent long code number precisely, ACHA mechanism use BigDecimal java class. The BigDecimal java class is a class that theoretically can keep unlimited mantissa value of a decimal number. Unfortunately, the decimal number represented by BigDecimal java class is encoded as string. Because of the limited number of bits contained in one SMS (1120 bits), there is only short java string that can be saved in one SMS.

To increase the number of BigDecimal code number value that can be saved in one SMS, the BigDecimal code number value should be represented more efficiently. In ACHA mechanism, to represent the code number, ACHA introduce base-100 number that can be encoded by using default GSM 7 bit alphabet. By the base-100 number introduced in ACHA, every two number of mantissa in BigDecimal code number will be converted to a base-100 number encoded in default GSM 7 bit alphabet. In ACHA, the converted BigDecimal code number is called with Encoded Code Number. For example, if number 91 in base-10 number is represented with Ü in base-100 number and number 23 in base-10 number is represented with M in base-100 number, then BigDecimal code number "0.9123" can be written as ÜM in compressed SMS. Note that either Ü or M can be encoded only by using 7 bits in default GSM 7-bit alphabet encoding system. By using introduced base-100 number encoded in default GSM 7 bit alphabet encoding system, ACHA mechanism can represent 2k digit mantissa of BigDecimal code number by only using k default GSM 7-bit characters.

Introduced base-100 number is also used to represent Conf_Char. Recall the discussion in section II.A, there are 2^n possible sub table configurations from n ACHA sub tables. Base-100 number is able to be used to represent all those possible configurations. The number of character needed to represent Conf_Char depends on the number of ACHA sub tables used. For example, if ACHA sub tables used are 6, then there are $2^6 = 64$ possible configurations. One base-100

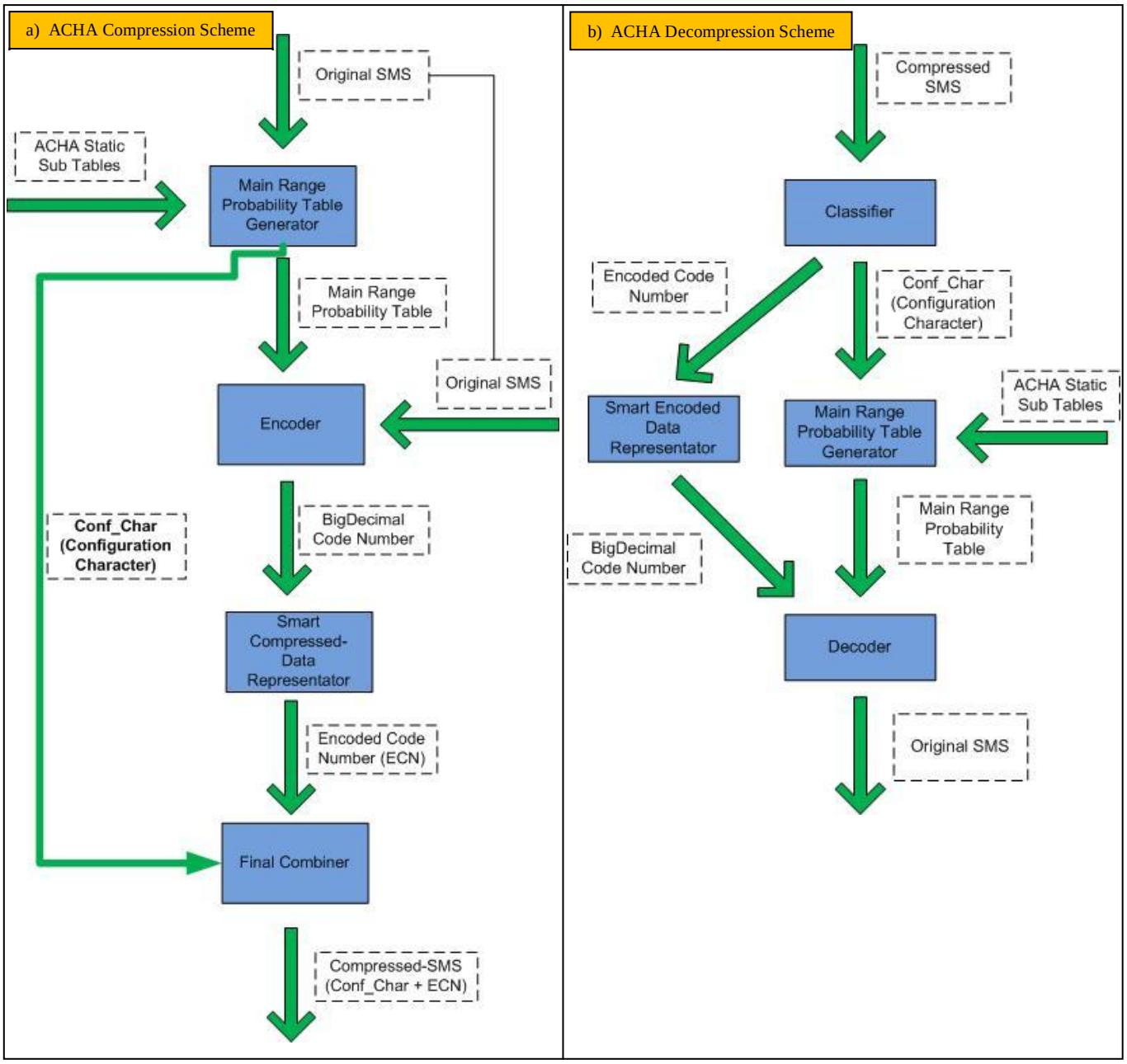

Fig. 1 ACHA Overall Process Scheme

number (one Conf_Char) is enough to represent all those possible configurations. Meanwhile, if ACHA sub tables used are 8, then there are $2^8 = 256$ possible configurations. Then it means that it needs at least two base-100 numbers (two Conf_Char) to represent all possible configurations. Because of this smart compressed-data representation, ACHA mechanism can represent any information of compressed-SMS especially the code number very efficiently. Fig 1 shows the overall process of ACHA mechanism.

III. Related Work

ACHA compression mechanism is adapted from hybrid compression mechanism using codebooks containing *k* static codes used by IBM's "Information Management System" (IMS) [2] –with some modified method elaboration. In IBM-IMS, there are *k* static compression codes to compress a particular character; meanwhile in ACHA there are *n* static sub tables used to build main conversion table (range probability table) to build compressed-data (code number). While in IBM-IMS the compression-code has been statically defined, in ACHA it is the sub table characters frequency that is statically defined. The compression-code in ACHA is always dynamic depends on the variety of characters in original data. If the characters-variety in original data changes, the ACHA sub tables needed to compress the original data will probably change too, thus the main range probability table will also change, as the result, the overall compression-code will be different too.

IV. Proposed Method Effectiveness

In ACHA, the overall compressed-SMS is written using standard default GSM 7-bit characters, it means that every character written in compressed-SMS can be read correctly by receiver phone (decoder), thus, there is no worry about unreadable characters in receiver phone. In ACHA mechanism, theoretically the more number of ACHA sub tables, the better ACHA compression ratio is. It happens because when number of ACHA sub tables increase, the number of unnecessary data frequency in main range probability table can be decreased. For example, if in Table 2 there are 16 sub tables (each sub tables consists of one single data frequency), it means to compress "ayah saya" there is only 5 data saved in the main range probability table. However, if the number of ACHA sub tables used increases, the number of Conf_Char needed will increase too. To find the best compression ratio of ACHA mechanism, obviously the best number of ACHA sub table used in ACHA mechanism must be found.

Because the proposed SMS-compression mechanism in this paper is quite new to compress SMS, the research to find the best number of ACHA sub table that should be used is still under observation. Although the best compression ratio of the proposed SMS-compression mechanism is still under observation, the effectiveness of the proposed mechanism has been tested well.

V. Implementation

To observe ACHA performance, the proposed mechanism has been implemented in mobile phone using Android operating system. The implementation has been done in Samsung Galaxy Tab (processor Cortex A-8 1GHz and 512MB of RAM). Fig 2 shows the interface of application implementing ACHA that is tested in Galaxy Tab.

Fig. 2 ACHA Application Interface

VI. Experiments and Results

The effectiveness of ACHA mechanism has been tested to compress 150 diverse SMS using Bahasa Indonesia. So far, the research has just tested the ACHA effectiveness using 19 ACHA sub tables to compress SMS using Bahasa Indonesia. Theoretically, ACHA mechanism can be used to compress any character in SMS effectively, however because the research about ACHA mechanism is a new research, on the research that has been done so far the variety of recognized character in encoder is just set to 96. Those characters are chosen based on the most 96 characters statistically used in SMS using Bahasa Indonesia. Fig 3 shows the effectiveness of ACHA mechanism compared to pure arithmetic coding that use dynamic range probability table.

Fig 3 shows that the average compression ratio of tested ACHA mechanism is about 71% (while the best ACHA compression ratio can reach less than 25%). The difference between ACHA average compression ratio and pure arithmetic average compression ratio (without considering the compressed-data header) is less than 6 %. Although ACHA mechanism compression ratio is not better than the pure arithmetic coding compression ratio (without compressed-data header), but practically with the existence of compressed-data header in pure arithmetic coding, the overall compression-

ratio of ACHA mechanism is proved to be better than pure arithmetic coding compression ratio. For such a small-size data like SMS, 71% compression ratio is a significant value.

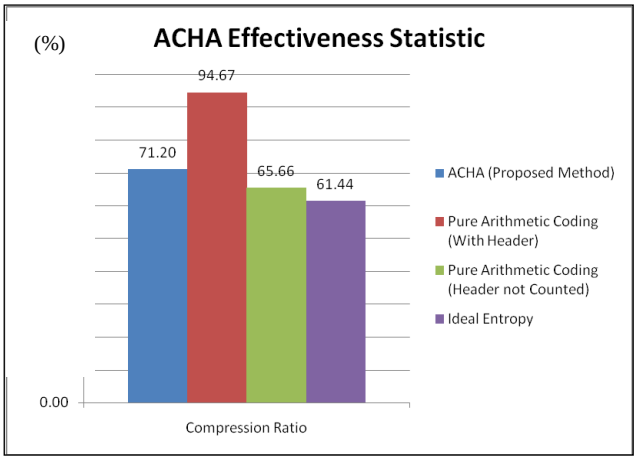

Fig 3. Average Compression Ratio of Varied Arithmetic Coding Compared to Ideal Entropy Compression Ratio

VII. Contribution to Society

As told in the previous section, the research about ACHA mechanism is still under observation. It means that the best ACHA compression ratio to compress any recognized characters in SMS possibly has not been found yet. A long with the continuance of the research, the best ACHA performance to compress SMS will be more revealed each time. When the best ACHA performance to compress any recognized characters in SMS is found, then there are two main optimal contributions that ACHA can offer to society: SMS bandwidth saving and possible-secured SMS data protection.

By compressing SMS body, the number of bits needed to represent the SMS body will be decreased. 200 characters that usually encoded with 1400 bits (more than 1 SMS maximum capacity) can be compressed to approximately 980 bits (less than 1 SMS maximum capacity) by using ACHA mechanism. Without being compressed, 200 characters SMS needs bandwidth to send 2 SMS. While by being compressed, the 200 characters SMS only needs bandwidth to send 1 SMS. This bandwidth saving obviously can be helpful for SMS provider resource efficiency. By saving the usage of SMS provider bandwidth, SMS provider service quality and the number of SMS-service user that can be served by SMS provider in a time will be increased.

Besides saving SMS bandwidth, ACHA can also be used as a part of encryption mechanism to protect SMS body. Table 5 shows compressed-SMS format of ACHA. Table 5 shows that the compressed-SMS practically can be treated as common characters in SMS. By implementing any trivial encryption method, for example Vigenere Cipher, the SMS body will be well-protected because no data frequency analysis can be done to crack the encrypted-data. Practically, other encryption methods can also be implemented to encrypt ACHA compressed-SMS. The main contribution of ACHA in protecting SMS is that it decreases significantly the possibility of data frequency analysis to crack encrypted-data.

TABLE 5
Compressed-SMS Example From ACHA Compression Mechansim

Original SMS	Compressed-SMS	Compression Ratio
Serpihan malam bulan separuh, jelang hari baru pertanda tiba. Kembang api membuncah ke angkasa sambut tahun muda 2011. Selamat tahun baru 2011 semoga lebih baik dibanding tahun sebelumnya.	pàogÜ*f3t_ck(;P2u 3ÜI/d1oU*PSVÇ! OIl/E@lTÜT?2?Nt 7v_+OH ùvä.Ürg'+tÜuugtüÅ åÇäN9ùQGéHuM* 0v'B,/Hrv3&W>é3 bés¿k>w@vjbYwL, 5/>0XXCa6öky/1ö v	68.62%
188 characters (2 SMS)	129 characters (1 SMS)	save 59 characters

When ACHA mechanism is tested in one of mobile phone using Android Operating System, i.e. Galaxy Tab (processor Cortex A-8 1GHz and 512MB of RAM), the ACHA mechanism tested to work properly. Fig 2 shows the interface of mobile application implementing the ACHA mechanism. On compressing 572 characters, in Galaxy Tab ACHA only spends less than 2 seconds. Although the best ACHA performance is still under observation, practically nowadays ACHA configuration has been able to be used to compress SMS effectively. Thus, both above main contribution that ACHA wants to offer practically can also be achieved by current ACHA configuration (although the contribution has not been optimal yet). Based on all the facts given previously in this paper, this paper claims that ACHA is SMS compression mechanism that practically efficient to compress SMS and it has significant positive contributions to society.

VIII. Conclusion

Based on research that has been done so far to ACHA mechanism, the proposed SMS compression mechanism proposed in this paper (ACHA: Arithmetic Coding Hybrid Ario) is proven to be practically effective to compress SMS body efficiently. ACHA mechanism has been tested on Android-operating-system-base mobile phone. The current average ACHA compression ratio that has been researched is 71%, while the best compression ratio of ACHA can reach less than 25%. However, this compression ratio is practically still not proven as the best ACHA compression ratio. Until today, the research to optimize ACHA compression ratio by finding the best ACHA sub tables configuration is still under observation.

Reference

[1] Witten, Ian H. Neal, Radford M. Cleary, John G. 1987. *Arithmetic Coding for Data Compression*.

[2] Lelewer, Debra A. Hirschberg, Daniel S. *Data Compression*.

[3] Mahmoud, Tarek M et al. *Hybrid Compression Encryption Technique for Securing SMS*.