

A Proposal of Substitute for Base85/64 – Base91

Dake He

School of Information Science & Technology, Southwest Jiaotong University, Chengdu 610031, China
College of Informatics, South China Agricultural University, Guangzhou 510642, China
dkhe_scce@swjtu.cn

Yu Sun, Zhen Jia, Xiuying Yu, Wei Guo, Wei He, Chao Qi

School of Information Science & Technology, Southwest Jiaotong University, Chengdu 610031, China

Xianhui Lu

Key Lab. of Information Security, Chinese Academy of Sciences, Beijing 100039, China

ABSTRACT

The coding transformation method, called Base91, is characterized by its output of 91 printable ASCII characters. Base91 has a higher encoding efficiency than Base85/64, and higher encoding rate than Base85. Besides, Base91 provides compatibility with any bit-length input sequence without additional filling declaration except for his codeword self. One can use Base91 as a substitute for Base85 and Base64 to get some benefits in restricted situations.

Keywords: Base91; Base85; Base64; printable ASCII characters; IPv6

1. BACKGROUND OF INVENTION

With the rapid development of Internet and its business application, unambiguous and unaffected transmission by net equipment has become more and more important. As all know that a net transmission using some of 95 printable ASCII characters, include space character, has this safe specific feature.

For example, unambiguous and unaffected transmission by net equipment is necessary for encrypting system (as PGP for Pretty Good Privacy) and E-mail. All the SMTP(Simple Mail Transfer Protocol)-based E-mail can provide compatibility with the E-mail. So-called compatibility with the E-mail is to transform arbitrary 8-bit data byte-strings or arbitrary bit stream data transferred by the E-mail into a character-strings of a limited ASCII (American Standard Code for Information Interchange). The main limitation on the latter is that: (a) the characters have to be printable; (b) the characters are

not control character or “-”(hyphen). There are totally 94 of such ASCII characters, their corresponding digital coding being all integers ranging from 32 through 126 with the exception of 45. E-mail written in these ASCII characters is compatible with the Internet standard SMTP, and can be transferred in nearly all the E-mail systems. Nowadays, as Content-Transfer-Encoding to provide compatibility with the E-mail, Base64^[1,2] code is usually employed.

Base64 coding divides the input sequence into blocks being 6-bits long to be used as variable implementation mapping, the mapping is denoted by

Base64[]: $X \rightarrow Y$

where the variable or original image set X includes all 64 6-bits long symbols (denoted as integers 0, 1, ..., 63) and Φ representing “no data” or empty , the image set Y includes the upper and lower cases of 26 alphabetic characters, Arabic digits ranging from 0 through 9, “+”, “/” and filling character “=”, where it is specified that in the non-program statements the Chinese quotation marks (“”) are used as the delimiter of characters or character-strings (the following is same). Mapping rules commonly used in Base64 coding software are

Base64[0]=“A”, ..., Base64[25]=“Z”, Base64[26]=“a”,
..., Base64[51]=“z”, Base64[52]=“0”, ..., Base64[61]=“9”,
Base64[62]=“+”, Base64[63]=“/”

Particularly, Base64[Φ]=“=” is used only when needed so as to make the total number of characters of output string equal to the multiples of 4. The coding efficiency of Base64 coding is $6/8 = 75\%$. The data expansion rate is $8/6 = 4/3 = 133.333\%$.

Another unambiguous and unaffected transmission by net equipment is Base85^[3]. Base85 (also called

"ASCII85") is a form of binary-to-text encoding developed by Paul E. Rutter for the btoa utility. By using five printable ASCII characters to represent four bytes of binary data, so the encoding efficiency of Base85 is 4/5 or 80%. The data expansion rate is 5/4 = 125%. Because always depending on modulo-85 division of four bytes, Base85 encoding rate is lower than Base64. Main modern use of Base85 is in Adobe's PostScript and Portable Document Format file formats, and in the compact representation of IPv6 addresses being 128-bits long.

2. BASE91 CODE

The aim of designing Base91 coding^[5,6] is to provide a new digital data coding method as substitute of Base85 or Base64, so as to get higher coding efficiency under the condition of E-mail compatibility, or more generally, of realizing the unambiguous and unaffected transmission by Internet equipment.

2.1 Base91 encoding

The main idea of Base91 is the new block design: to divide the input data sequence into blocks of 13-bits long, and to encode one block as two printable ASCII characters of Base91. Why 91? Because

$$90 \times 90 < 2^{13} (=8192) < 91 \times 91 (=8281),$$

so 91 is the best choice of basis or radix number in this case. Base91 encoding mapping is denoted by

$$\text{Base91}[\]: X \rightarrow Y$$

where the variable or original image set X includes all 8192 13-bits symbols (denoted as integers 0,1,...,8191) and symbols $\varphi_n=8191+n$ ($n=1,\dots,12$) being filling data declaration denoting the n-bit data at the specified side of the last block are filling data, thereby making the total number of elements in the original image set equal to 8204; the image set Y is the sub-set of the direct product of $R91 \times R91$, where the symbol R91 denotes the set of 91 characters selected from the 95 printable ASCII character set with "-", "=", "." and space characters excluded, the direct product $R91 \times R91$ has 8281 elements. So, the remained 77 elements of Y may be used to extend Base91 for the future.

Base91 is defined as an injective mapping arbitrarily selected from X into the direct product $R91 \times R91$. For the convenience of implementation, assuming that $R91_ch[91]$ is the character array that includes all R91

characters and is arranged according to the ASCII sequential order, we preferably selects the following mapping:

$$\begin{aligned} \text{Base91}[x] &= (\text{ch1}, \text{ch2}) \\ &= (R91_ch[x/91], R91_ch[x\%91]) \end{aligned} \quad (1)$$

where $x \in X$, $\text{ch1}, \text{ch2} \in R91$, symbols "/" and "%" are the operators used in the C language, representing integral division and modulo division (remainder) respectively. The operation of dividing the input message into 13-bits long blocks may produce the last block less than 13-bits long. For such block, n bits are added to the specified side to make it become a complete block for encoding mapping, and a block of data φ_n is added thereafter as the input data implementing mapping so that it can be decided how many filling bits have to be deleted during decoding. When needed, double-character "=" may be used as a "terminating symbol" of the output character string. Hence at most 92 printable ASCII characters can appear in the output-string of Base91 encoding.

According to the encoding rules of the above-mentioned Base91, the number of extra added output data consisting of the filling bits, the image of the filling data declaration φ_n and the "terminating symbol" does not exceed 6 characters. Therefore, with the increase of the length of input data sequences, the encoding efficiency of Base91 approaches 81.25%, its data expansion rate approaches 123.077%.

Compared with the Base64 or Base85, the Base91 has its advantage in encoding efficiency. The design features of the three coding schemes are shown in table 1.

Table 1 design features of three code schemes

Code scheme Features	Base64	Base85	Base91
basic characters in output	64	85	91
encoded bits in output B	6	6.4	6.5
input bits (one block)	6	32	13
output bits (one block)	8	40	16
encoding efficiency	75%	80%	81.25%
data expansion rate	133.333%	125%	123.077%
codeword for filling bits	0	0	12

2.2 Implementation of Base91 encoding

There is some technique to speed up encoding of

Base64 and Base91. But the technique of lookup KB-table cannot be used to Base85. Noting that Base85 encoding mapping may denoted by

$$\text{Base85}[x] = (\text{ch1}, \text{ch2}, \text{ch3}, \text{ch4}, \text{ch5})$$

for $(i=5; i>0; i--)\{\text{chi} = \text{R85_ch}[x\%85]; x = x/85;\}$ (2)

where x is a 32-bits input, $\text{chi} \in \text{R85}$, and $\text{R85_ch}[85]$ is the character array that includes all R85 characters and is arranged according to the ASCII sequential order, symbols “/” and “%” are the operators used in the C language, representing integral division and modulo division (remainder) respectively. Because the block being 32 bits and the divisor being 85, Base85 has to repeatedly employ normal integer division of 32 bits long words. In contrast with Base85, the operators “/” and “%” only rely on 16 bits words in normal encoding transformation of Base91.

So, at any case Base91 always has a higher encoding rate than Base85. Using normal encoding transformation of operators “/” and “%”, Base91 has much lower encoding rate than Base64. If using lookup KB-table technique to speed up, then two encoding rates of Base64 and Base91 are similar, but the table for Base91 is bigger than Base64.

Table 2 40MB encoding time//rate of Base64/91/85

Code scheme	Base64	Base91	Base85
Intel Celeron 330, 2.66GHz, L1 data 16KB with MMX, SSE, SSE2,SSE3, Compiled C			
Alg. without lookup KB's table	161318 // 248MB/s	359262 // 111MB/S	812440 // 49.3MB/s
Alg. with lookup KB-table ignore tab. loading time	90784 (not opt.)// 440MB/s	84214 // 474MB/s	
Intel Celeron 2.5GHz Compiled C : GCC [4]			
Alg. with 6KB~8096B lookup table	//306MB/s : 453MB/s		
IBMX40 P M738, 1.40GHz, L1 data 32KB with MMX,SSE,SSE2, Compiled C			
Alg. with lookup KB-table ignore tab. loading time	104264 (not opt.)// 384MB/s	99867 // 400MB/s	
Intel P4, 2.80GHz, L1 data 16KB with MMX, SSE, SSE2,SSE3, Compiled C			
Alg. with lookup KB-table ignore tab. loading time	52242 (not opt.)// 766MB/s	51381 // 778MB/s	

Many tests of encoding and decoding rate of Base91, Base64, Base85 have made by us. In our tests the input

data are 40MB (average over en/decoding 100 times), the unit of expend time is μs , and the unit of en/decoding rate is MB/s. Because there are many differences of hardware and software, we just give the timing results and relative parameters in our tests in Table 2 and Table 3. The timing result of Base64 reported by reference [4], which is a little better than ours of Base64 and Base91 because that programmer made more things of optimum programming in a PC with a little poor hardware than ours, we think and don't know his timing result whether including 8096-byte table loading time or not, is listed in Table 2, too.

Table 3 40MB en/decoding time & rate of Base64/91/85

Code scheme	Base64	Base91	Base85
Intel Celeron 330, 2.66GHz, L1 data 16KB with MMX, SSE,SSE2,SSE3, Algorithm without lookup KB's table to speed up, Compiled C			
encoding time	161318	359262	812440
decoding time	125210	135538	051841
en+decoding time	286528	494800	864281
average rate of encoding & decoding	279MB/s	161MB/s	93MB/s

3. USAGE OF VASE91 IN THE COMPACT REPRESENTATION OF IPv6 ADDRESSES

Base85 is used in the compact representation of IPv6 address being 128 bits long, the result is 20 printable ASCII characters^[3]. By a 0.1 increase of encoded bits in one output byte, Base91 can do same things, but a note of usage is needful.

We have two concrete way: 1) to divide IPv6 address into one block of 11-bits and nine blocks of 13-bits; 2) to divide IPv6 address into four 32bit-blocks. In the first case encoding is ordinary, but don't use special codeword φ_n because the bit-length of 128, the output of decoding, is known. In the second case a note of usage is following.

Differently from Base85 repeatedly using normal integer division of 32-bit long words, we divide 32-bit block into three sub-blocks (from left): B1 being 6-bits, B2 and B3 being 13-bits respectively. The encoding of B2 and B3 is general as Eq.(1) and gives output bytes of C2,C3,C4,C5. But for B1, we can directly encode it as

$$C1=R91_ch[B1] \quad (3)$$

because 6bit-B1 is less than 91. For the inverse transformation, which maps C1,C2,C3,C4,C5 to 32bit-block, the rate of Base91 decoding is similar to that one of Base85 decoding in Table 3, because there needs no division operation.

4. CONCLUSION

Base91 has a higher encoding efficiency than Base64 or Base85, and has a higher encoding rate or (encoding+ decoding)'s rate than Base85. In the case of CPU with 16bit-word, Base91 has more superiority over Base85. If using lookup KB-table to speed up, then encoding rate of Base91 is similar to one of Base64. Besides, Base91 provides compatibility with any bit-length input sequence without additional filling declaration except for his codeword self.

So, we suggest that Base91 is good substitute for Base85 firstly, and maybe is qualified substitute for Base64 to get some benefits in restricted situations. Recalling the difference of data expansion rates of Base91 and Base64, see Table 1, there is 7.7% (of Base64's encoded data) cut off in data traffic, transfer time, transfer cost, at the same time there is saving our society some energy, if employing Base91 instead of Base64 as encoding for network data transfer with huge data.

5. REFERENCES

- [1] RFC4648, "The Base16, Base32, Base64 Data Encoding", 2006.
- [2] RFC2045, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", 1996.
- [3] RFC1924, "A Compact Representation of IPv6 Addresses", 1996.
- [4] http://www.experts-exchange.com/Programming/Languages/CPP/Q_21988706.html
- [5] A digital data transforming method, Chinese Patent ZL00112884.1, Application date: Apr.28,2000, inventors: Dake He, Wei He.
- [6] Digital Data Transforming Method, US 6,859, 151B2, Feb.22,2005, inventors: Dake He, Wei He.