Multilevel Security with Restricted Character Set Translation

Chyan Yang
Dept. of Electrical & Computer Eng.
U.S. Naval Postgraduate School
Monterey, CA 93943

Chien-Chao Tsai
Military Integrated Comm. Agency
Ministry of Defense
Taipei, Taiwan R.O.C.

Abstract

The Navy message traffic is transmitted with restricted character set and optionally the files are compressed. Both character translation and data compression can be used as add-on data encryption. This paper presents the restricted character set translation algorithms. The algorithms has been implemented with multilevel security access control using master key scheme and have been incorporated in a data compression program for military applications.

1 Introduction

Since Naval message traffic uses restricted character set listed in NTP3 Annex C [1], the compressed and/or encrypted package has to be translated using this restricted character set. A restricted character set of N symbols can be represented as $C = \{\alpha_1, \alpha_2, \dots, \alpha_N\}$, where $N \leq 256$.

The Naval message traffic allows only 45 restricted characters. Apparently, there must be some data expansion in the character translation process. However, the electrical update to the fleet via existing communications channels makes this a must. Note that if the character translation algorithm is unknown, the translation itself can serve as an additional encryption. In the following sections we will analyze the expansion ratios as well as the software implementation regarding the restricted character set translation.

2 Restrict Character Set Translation

Without loss of generality, it is assumed that the restricted 45 characters are in the contiguous decimal value range [46, 90] (ASCII '.' to 'Z'). Actually the characters '<' and '>' are not used in NTP3; instead, the characters '{' and '}' are used. In a source

package, without character set translation, each input byte of 8 bits can assume $2^8 = 256$ various bit patterns and all patterns are equally likely to occur. When mapping a byte of 8-bit to one of the 45 restricted character, one may let 40 bit patterns uniquely map to corresponding 40 characters; consequently the mappings of the other 216 patterns have to use 2 characters each. In other words, five characters are reserved and used as leading characters for mapping bit patterns to 2-character pairs; these 5 leading characters can accommodate $5 \times 44 = 220$ bit patterns. On the average, in using this method, the translated file is expanded to 185% of the original file since $(40/256) \times 1 + (216/256) \times 2 = 1.84375 \rightarrow 184.37\%$.

The Navy requires expansion ratios not larger than 50% [2]. The expansion ratio of 85% is unacceptably high therefore a source file must be translated in blocks of bits smaller than 8 when data storage efficiency is concerned. Similarly, one can verify that if the translation is performed in 7-bit blocks by assigning 43 bit patterns to single characters and other 85 bit patterns to pairs of two characters, one would get expansion ratio of 90%.

Note that $5 < \log_2(45) < 6$. Thus, the bit pattern to be translated could be blocks of either 5-bit or 6-bit depending on the efficiency of expansion ratio to be discussed below. Namely, a translated character (8 bits) can represent a block of either 5-bit or 6-bit of input bit stream. This implies that an output byte (character) always starts with a '0' bit and the other 7 bits varies in 45 patterns. Three basic translation methods are discussed as follows:

1. Method A: Scan input stream in 6-bit blocks. Since a 6-bit block may form 64 different patterns, with only 45 characters to map there are 44 lucky 6-bit patterns that can map to a single character in the restricted character set (say, $\alpha_1, \alpha_2, \dots, \alpha_{44}$) whereas the rest of 20 patterns has to be translated in two combined characters $\alpha_{45}\alpha_i, i = 1, \dots, 20$.

29.5.1

2. Method B: Scan input stream in 5-bit blocks. Since a 5-bit block may span 32 different patterns, with 45 characters to map there are 13 characters in the restricted character set unused.
3. Method C: This is an improvement to the second method and will translate 6-bit block whenever possible. It scans the input stream, however, in 6-bit blocks before committing to a translation. We may assign 32 restricted characters to integer values [0, 31] for 5-bit blocks and the other unused 13 characters to [32, 44] for 6-bit blocks. If the value of the 6-bit block is in the range of [32, 44], then the block is translated to the corresponding character. When the value of a 6-bit block is not in the range of [32, 44] then it is either in [0, 31] or in [45, 63]. The algorithm shifts one bit backward (unget) leaving the value in [0, 31] and translates the 5-bit block. In the following discussion let S denote the integer interval [32, 44].

2.1 Expansion Ratios

It can be shown that the first two methods, A and B, are not as efficient as the third method. Without a prior knowledge of the source stream, it is reasonable to assume that all bit patterns are equally likely to occur in the following discussions. Let b_1, b_2 be the number of bits used to encode a translated character (byte) and P_{b1}, P_{b2} be the corresponding probabilities of occurrences in translation. The expansion ratio η of Method A is then

$$\eta = (8 - b_1)P_{b1}/b_1 + (8 \times 2 - b_2)(1 - P_{b1})/b_2 = 0.75$$

where $b_1 = 6, P_{b1} = 44/64 \approx 0.688, b_2 = 6, P_{b2} = 1 - P_{b1} = 0.312$. In the above computation, the second term indicates that we expand from 6 bits to 2 bytes for the 20 unlucky 6-bit patterns. Similarly, we can compute the expansion ratio for the Method B as $\eta = (8 - 5)/5 = 0.6$ or 60%. For the Method C, $b_1 = 6, b_2 = 5$, thus $\eta = 0.546$. The third method provides the best of all three translation methods and is very close to the set Naval requirement of 50%.

Variants of the Method C can increase the probability of 6-bit block pattern translation. But, they are not as efficient as the method C. For example, when $N = 45$, one may assign 16 characters for 4-bit and 29 characters for 6-bit. Two other variants are (1) 8 characters for 3-bit and 37 characters for 6-bit and (2) 4 characters for 2-bit and 41 characters for 6-bit; their corresponding expansion ratios are 0.698 and 1.292. All three of these results are worse than Method C because the second term in each calculation grows faster

than the reduction of the corresponding probabilities. Moreover, it is impossible to translate more than 6 bits in each decision when $N < 64$.

To generalize the η computation of Method C for restricted character set of size N in the range [2, 256], η can be calculated as follows:

$$\eta = (8 - b_1) \times (P_{b1}/b_1) + (8 - b_2) \times (P_{b2}/b_2)$$

where

$$b_1 = \lceil \log_2(N) \rceil, P_{b1} = \frac{N - 2^{\lceil \log_2(N) \rceil}}{2^{\lceil \log_2(N) \rceil}} = \frac{N - 2^{b_1}}{2^{b_1}}$$

and $b_2 = \lfloor \log_2(N) \rfloor, P_{b2} = 1 - P_{b1}$.

The equation here is similar to the computations used in previous paragraph except that it is now parameterized with N . The plot of this equation, expansion ratios vs. character set size, declines sharply before N reaches 16 [4]. It seems that when N is in the range of [32, 64] the expansion ratios are quite satisfactory. Nevertheless, practical operation environment dictates the choice of N . For example, in order to accommodate a set of Morse code communication, the choice of $N = 45$ seems reasonable regardless of expansion ratio.

Let 'PAK' represent the file types used by Naval data package which consists of 90% ASCII text and 10% binary or image data. An extensive performance analysis on data compression software found that an average of 39.3% compression ratio is achievable [3]. Theoretically, by taking the expansion ratio 54.6% of Method C above with an average compression ratio around 39.3%, the compressed and translated file size would be about $39.3\% \times (1 + 54.6\%) = 60.77\%$ of original file size. Representative testing results of different type of files after compression and character set translation [3, 4] show that the average expansion ratio is 60.4%: fairly close to theoretical ratio (60.77%) described above. This meets the requirement set by the Naval Security Group Detachment (NSGD) [2].

2.2 Software implementation

A C program based on Method C was implemented [1]. We now describe the algorithm that has been incorporated in a compressed/encrypted data package. The character set translation algorithm has two separate parts: translation (at host system) and recovery (at remote system).

2.2.1 Translation Algorithm

We may assign 32 restricted characters ('.' through 'M') to decimal values interval [0, 31] for 5-bit blocks

and the other 13 characters ('N' through 'Z') to the interval [32, 44] for 6-bit blocks. If the value of the 6-bit block is in the interval [32, 44], then the block is translated to the corresponding character. When the value is not in the range of [32, 44] then the algorithm shifts one bit backward (unget) making the value reside in interval [0, 31] and translates the 5-bit block.

Refer to the bit pattern dissection of a translation example below, when the input string is coming from right to left, we observe the following:

```
10011000  11011110  10010001
<--1-->
      <---2--->
            <---3--->
                  <--4-->
```

- Bit pattern 1 ('100110', the LSB is 0): Decimal value = $38 \in S$, translates the 6-bit block and output $38 + 46$ ('T'). Here the 6-bit block of 100110 has binary value 38 which can be found in Table 1.
- Bit pattern 2 ('001101'): Decimal value = $13 \notin S$, translates 5-bit block of '00110' (shift the pointer 1 bit backward, i.e., to left), and output $6 + 46$ ('4'). Note that 00110 has a binary value 6.
- Bit pattern 3 ('111101'): Decimal value = $61 \notin S$, translates 5-bit block of '11110', and output 'L'.
- Bit pattern 4 ('100100'): Decimal value = $36 \in S$, output $36 + 46$ ('R').

The displacement of 46 above maps decimal values into the desired ASCII code range ['.', 'Z']. The output characters will be 'T4LR....' and leave the last 2 bits to be the more significant bits of the next 6-bit block. When the EOF or last byte of buffer is encountered, the remaining bits will be padded with 0s in LSB to form the last pattern. For the above example, if '10010001' is the last input byte, then the last 6-bit pattern will be '010000' and is translated to '01000' + 46 ('6'); the output is then 'T4LR6'.

2.2.2 Recovery Algorithm

The displacement of 46 made in translation has to be reset for each input character in recovery at the receiving hosts. If the value after reset is in S , an original 6-bit translated pattern is assumed and a 6-bit block is recovered; otherwise it maps to a 5-bit block. The output characters 'T4LR6' in the previous example will be recovered to the original bit string.

The process is explained now. First, the bit pattern of 'T4LR6' can be represented as hexadecimal string '54 34 4C 52 36'. After a 46 offset taken on each byte, the binary values corresponding to these five bytes become '38 6 30 36 8'. In other words, the input to the inner loop of the recovery routine is now '00100110 00000110 00011110 00100100 00001000'. This bit string will be recovered as '10011000 11011110 10010001'. Because the file before translation is byte-oriented, the recovery of the last input character should complete the last byte of original compressed/encrypted package.

3 Improvement by Patterns Reassignment

In this section, each bit pattern corresponding to output character will be examined. It will be shown that one can further reduce the translation expansion ratio by suitable reassignment of bit patterns.

3.1 Unused Patterns in Translation

The translation algorithm discussed in previous sections examines an input 6-bit block and translates either 6 bits or 5 bits of the input blocks. Theoretically, when each input pattern is assumed to be equally likely to occur, having been compressed and/or encrypted, the Method C with expansion ratio 54.6% seems to be an optimized algorithm. Having enumerated all patterns, however, one can further reduce the expansion ratio to less than 50% — the Navy requirement [2].

The clue is that certain 5-bit patterns do not appear in practical translation procedure due to the 1 bit shift operation (unget) when the 6-bit value is not in S or interval [32, 44]. Table 1 lists all 6-bit patterns with corresponding decimal values and output characters. Notice that in Table 1 the 5-bit blocks in sets S_L and S_U exhibit redundancies and six patterns do not occur (values in interval [16, 21]). For instance, the first two 5-bit blocks in S_L are both '00000'. Note that of the 6-bit patterns in S_L and S_U only 5-bit values are actually translated while each of the rightmost bit is restored back to input stream. Let x denote a don't care bit. The 6 missing patterns are '10000 x ', '10001 x ', '10010 x ', '10011 x ', '10100 x ', and '10101 x ' with corresponding characters '>', '?', '@', 'A', 'B', 'C' respectively. In other words, when Method C in Section 2 is used to translate 64 6-bit patterns, only 39 (45 - 6) characters are actually assigned with 25 of them appearing twice and leaving 6 characters unused. The effort now is to translate patterns in S_L or

Set	6-bit Patterns	5-bit* Value	6-bit Value	Output**
S_L	000000	0		'.'
	000001	0		'.'
	000010	1		'/'
		..		...
		..		...
	011101	14		'<'
S	011110	15		'='
	011111	15		'='
	100000		32	'N'***
	100001		33	'O'
		..	..	...
		..	..	...
S_U	101011		43	'Y'
	101100		44	'Z'
	101101	22		'D'
	101110	23		'E'
	101111	23		'E'
		..		...
S_V	111110	31		'M'
	111111	31		'M'

Table 1: List of 6-bit Patterns and Their Output Characters; * When unset 1 bit, ** Total 39 characters, *** 'N'=32+46.

S_U in 6-bit blocks by assigning unused characters to them.

3.2 Characters Reassignment

Using the 6 unused characters can improve the expansion ratio. These 6 unused characters may be assigned to the first six unique 6-bit blocks of S_U (from '101101' to '110010', see Table 1). That is, we can assign these 6 characters to values in [45, 50]. By doing this, the characters originally assigned to interval [45, 50] (4 characters : 'D', 'E', 'F', and 'G') become unused. These four characters can be used to substitute another four 6-bit patterns, say [51, 54] (from '110011' to '110110'). Furthermore, the characters 'H' and 'I' correspond to [51, 54] are reassigned to [55, 56]. This recursive characters reassignment may continue until value 57 is assigned and there are no more unused characters; $6+4+2+1=13$ characters has been reassigned. As shown in Table 2 through appropriate displacement of +46 (within S_A), +30 (within S'), or +59 (within S_B) we can rearrange all output characters to be contiguous similar to that in ASCII code. The 6-bit patterns '100000' through '111001' (interval

Set	6-bit Patterns	5-bit Value	6-bit Value	Output
S_A		..	0	...
		..	...	...
	011111	15	31	'='
S'	100000		32	'>'
	100001		33	'?'
		..	..	...
	101011		43	'I'
	101100		44	'J'
		..	..	...
S_B	111010		57	'W'
	111010	29		'X'
	111011	29		'X'
		..	..	...
	111110	31		'Z'
	111111	31		'Z'

Table 2: List of 6-bit patterns with new assignments

[32, 57]) is now assigned to characters '>' through 'W'. All other 6-bit blocks (in S_A or S_B) still have to be translated in 5-bit block.

3.3 Expansion ratio improvement

The expansion ratio is improved because we increase the probability of translating 6-bit blocks and reduce that of 5-bit blocks. For all 64 possible 6-bit patterns, we now have 26 patterns that can be translated in 6-bit blocks and 38 patterns have to be translated in 5-bit blocks. The overall expansion ratio is then : $\eta = (8-6)/6 \times (26/64) + (8-5)/5 \times (38/64) = 0.4917$. This expansion ratio shows a 5.43% (= 54.6% - 49.17%) improvement over that of the Method C and achieves the expectation set by Navy Security Group.

3.4 Experiments

The text listed below shows a short ASCII file that is to be processed through data compression, data encryption, and character set translation.

I was very pleased to receive your draft data compression research proposal dated February 27th. As requested, I am enclosing a detail list of our projected requirements to further assist you in smoothing your proposal and formalizing thesis work on this subject.

File	Size (byte)	Translation Size(η)	Comp& Trans. SIZE(%ORG)
Text	24969	37243(49.16%)	13853(55.48%)
WP	25195	37570(49.12%)	15082(59.86%)
C	17325	25864(49.29%)	6703(38.69%)
Binary	24630	36745(49.19%)	19695(79.90%)
PAK	76644	114625(49.56%)	44626(58.23%)
PAK1	39024	58327(49.46%)	23327(59.78%)
PAK2	104622	156516(49.60%)	61662(58.94%)
PAK3	174226	260362(49.44%)	95402(54.76%)

Table 3: Testing Results of Improved Translation; WP: WordPerfect, C: C Source, PAK: Navy Data Package.

The compressed and encrypted file after character set translation is shown below; all characters appeared within '.' and 'Z' as desired. Note that, there is neither CR (carriage return) nor LF (line feed). The file is displayed in multiple lines for readability. Because the original file is small and therefore resistant to compression, translated file size is larger than that of original file.

```
B2C8/3>:O.F4./C6:B./55XW;CP.2/A>2N63/24160.27
043.H;A<R6D;;JY.8/?22R7QZ;3M<>N74.IP;TO<7.XJK
3:T3P6L9:PPALJT6>X4Z.88;MT;QP;4F4Z29OMX=3.8BH
Q<90Q<B8<X:9G?6455U;QT<YRY5:UXHU;NR8B.=ZP91J9
.UZ7;YO;U;LW:5:OB=ZWLK3;94W<D160XWBTX4?<N641I
1XEASDOMXM5Y3LV<TR<Z:LXAD82XZRXR=<E=56<VPT7SD
=YQ7F1MB1=A8OP2M2/<WH>;:Y7EDD2LNUY8:<N=SY8UD
==6K;98LZ=OZ6N6ZT.5:4J
```

The theoretical compressed and translated file size can be reduced to : $39.3\% \times (1 + 49.17\%) = 58.62\%$. The results of a set of files are listed in Table 3. Expansion ratios in Table 3 are consistent with the predicted 58.62%. The PAK files in "Comp&Trans." column are reduced to 58.23%, 59.78%, 58.94%, or 54.76% of original file sizes; these are very close to the theoretical value 58.62%!

The discussion in this section assumed that the occurrences of all bit patterns are equally likely in the input to the translation algorithm. This is a reasonable assumption since the input bit patterns are dictated by the pointer values in unknown compression/encryption algorithms. When bit patterns are not equally likely the translation algorithm may be more sophisticated but may achieve a better expansion ratio. For example, if a text file to be processed

does not require compression or encryption, the expansion ratio of the character set translation should be smaller than the theoretical 49.17% because the first '0' bit and/or the second '0' bit of each input byte may be skipped in the translation process.

4 Conclusion

Various schemes of restricted character set translation have been investigated and their implementations on computers are discussed. The restricted character set has been utilized to defer hardware changes needed to expand the transmittable character set. Software can be used to expand the restricted character set without the need to buy expensive hardware.

Acknowledgments

This research has benefited tremendously from the Naval Security Group Detachment, Pensacola, Florida. The authors are grateful to Captain Engel and his staff for the initiation of the research subject. Moreover, the authors are indebted to Professor Paul Moose and U. S. Navy Lt. Phong Nguyen for helpful comments on our working draft.

References

1. Commander, Naval Computer and Telecommunication Command, *NTP3, Annex C*, 4401 Massachusetts Av. NW, Washington DC 20394-5000. Unclassified but need to know basis.
2. Commander, Naval Security Group Detachment, Corry Station, *Data Compression Project Specification*, March, 1991.
3. Y. J. Jung, C. Yang, and G. A. Myers, "Performance Analysis of Data Compression and Archiving Software for U. S. Navy," 9th Annual Decision Aids Conference, June, 1992.
4. C. C. Tsai, "Multilevel Security with Master Key and Restricted Character Set Translation in Data Compression", Naval Postgraduate School, Master Thesis, Mar. 1992.