Using SAS® Bitwise Functions to Scramble Data Fields with Key

Sheng Luo, Providian Corporation, Frazer, PA
Xinsheng Lin, IMS America, Plymouth Meeting, PA

Abstract

The data modeling in banks or insurance companies often requires data files to travel from or to other companies. Some data fields, such as credit card numbers and social security numbers, while necessary to be included in the file, need to be unknown to other companies. Encryption is a way to do this by replacing original characters in the file by some other characters that make no sense. Annette I. Ladan had an excellent discussion of the keyless encryption algorithms in SAS, at the SUGI 20 Conference. This paper tries to extend the subject further to algorithms of the keyed encryption in SAS. A special effort is made to discuss bitwise operations and SAS newly added bitwise functions, as well as the pseudo bitwise operations that authors used before SAS bitwise functions came out. A sample SAS program and its result are also displayed. Although the discussion is based on results from UNIX system and ASCII data, the algorithms and sample SAS program can be applied to any operating system and any type of data.

Introduction to the Keyed Encryption

To perform data modeling in a large direct marketing company, it is very common for a large number of text files to flow in and out of the company in order to overlay some external variables. Some customer data fields, such as credit card numbers, social security numbers and customer's names are not for the other company who appends data to know. Of course, the other company will not know if these data fields are excluded from the file. But we often cannot exclude them because we need them for matching overlaid data back to our own databases. The encryption is one way to protect these sensitive data fields from public. What the encryption does is to transfer original data values to other values that look meaningless. When the file comes back from the other company, we can decode scrambled data fields back to their original values. There are many ways to do the encryption. It can be keyed or keyless. The keyless encryption algorithms are conceptually simple in general, involving changing positions of data values or changing one character to another on one to one basis. Annette Ladan's paper presented at the SUGI 20 Conference provided an excellent discussion on algorithms of the keyless encryption in SAS. The keyed encryption algorithms can be conceptually complex, in the ways of linking original characters and keys (or password) by some mathematical expression. There could be many kinds of the keyed encryption algorithms. However, a good one should be able to encrypt or decrypt the data values by the same program, which is hard to do in a keyless encryption algorithm. A good one is also able to replace the same original character to different ones so that the scrambling pattern is not obvious.

Many good algorithms of the keyed encryption utilize bitwise operations, such as bitwise OR, bitwise XOR(eXclusive OR), and bitwise AND. Unlike regular Boolean comparisons that always results in either 0 or 1, the bitwise operations compare the each pair of corresponding bits of two values independently. The result can be any integer. For example, in the ASCII collating sequence, letter 'A' bitwise OR letter 'Z' is equal to the character '['. The calculation goes like this; 'A' is equal to 65 in the ASCII decimal and 01000001 in the ASCII binary; 'Z' equals to 90 in the ASCII decimal and 01011010 in the ASCII binary. 'A' bitwise OR 'Z' looks like:

```
 A : 01000001
 Z : 01011010
-----------------
OR: 01011011
```

The resulted 01011011 equals to decimal 91 which represent the character '[' in ASCII collating sequence. Likewise, 'A' bitwise XOR 'Z' equals to 27 in ASCII decimal; 'A' bitwise AND 'Z' equals to decimal 64 which corresponds to the character '@'. The bitwise operations also include shifting bits to the right or to the left, and complementing each bit.

Among all bitwise operations, the XOR is most important for encryption. When performing XOR between two bits, the result is one only if sum of two bits is one. Otherwise, the result is zero. Any value

XORed by the second value, or the key, resulting in the third value, can be restored by performing XOR between the key and the third value. For example, the letter 'A' bitwise XOR the character 'z' equals to the character ';' as shown below:

```
'A' : 01000001
'z' : 01111010
-----------------
';': 00111011
```

The character 'z' here serves as the key. When the character ';' bitwise XORs the character 'z', the original letter 'A' will be brought back. Therefore, by performing bitwise XOR, any character can be changed to other and restored back from other.

Other bitwise operations, such as bitwise AND, bitwise OR, are also used in encryption expression sometimes. However, they only serves to restrict results to certain characters. Often their use can be replaced by regular arithmetic expressions to achieve the same results.

A good example of the keyed encryption algorithms (see reference on Robert Alonso) is as follows:

Let's denote:
C{i} as ASCII value of the ith character of the data field to be scrambled; and
P{k} as ASCII value of the kth character in the password, or keys; and
OR, XOR and AND as birwise OR, bitwise XOR and bitwise AND respectively unless specified otherwise. Then

Step 1: X{i}=C{i} XOR P{k};
Step 2: Y{i}=[254-(254 AND i] OR (127 XOR i);
Step 3: Z{i}=X{i} XOR Y{i};

Thus Z{i} is scrambled value of C{i}. If we exchange the positions of Z{i} and C{i} in the algorithm, we can decrypt Z{i} to C{i}. The expression in the second step is to limit values of Z(i) between 128 and 255 and to further complicate pattern of Z(i). Y(i) can be also considered as another key.

Before the new bitwise functions were added to SAS, it was difficult to perform keyed encryption as described. Authors used PUT, INPUT, TRANSLATE, BYTE, INT, MOD functions, $BINARY format and BINARY informat to perform bitwise operations for scrambling data fields in SAS. Now with new bitwise functions such as BXOR, BOR, and BAND, the encryption is made much easier in SAS. The next section discuss in detail how to encrypt data fields in SAS.

Encryption by Bitwise Operations in SAS

The newly added bitwise functions in SAS has greatly improved efficiency of codes for encryption in SAS. To illustrate, consider author's old pseudo XOR, OR and AND operations between the letters 'A' and 'Z'.

First, we transfer these two letters to binary representation by PUT function and $BINARY format:

```
A_BINARY=PUT('A',$BINARY8.);
Z_BINARY=PUT('Z',$BINARY8.);
```

The value of A_BINARY is '01000001'; and value of Z_BINARY is '01011010'. They are both character variables.

Next, we sum up these two variables in $BINARY format by coding:

```
AZ=A_BINARY+Z_BINARY;
```

The value of numerical variable AZ is 2011011. Then we can perform AND, OR and XOR between 'A' and 'Z' by TRANSLATE function:

```
/* A AND Z by  2s to 1s and 1s to 0s */
A_AND_Z=TRANSLATE( LEFT(AZ),'01','12');

/* A OR Z by replacing 2s to 1s */
A_OR_Z=TRANSLATE(LEFT(AZ),'1','2');

/* 'A' XOR 'Z' by replacing 2s to 0s */
A_XOR_Z=TRANSLATE(LEFT(AZ),'0','2');
```

Remember the value of AZ is 2011011. Thus the values of A_AND_Z, A_OR_Z and A_XOR_Z are '1000000', '1011011' and '0011011' respectively.

To show the character representation of A_AND_Z, A_OR_Z and A_XOR_Z, we use BYTE and INPUT functions and BINARY informat in SAS:

```
/* RETURNS '@' */
C_AND=BYTE(INPUT(A_AND_Z,BINARY8.));

/* RETURNS '[' */
C_OR=BYTE(INPUT(A_OR_Z,BINARY8.));

/* RETURNS A FUNNY CHARACTER RELATED TO
ASCII CODE 27 */
C_XOR=BYTE(INPUT(A_XOR_Z,BINARY8.));
```

With SAS new bitwise functions, the above operations can be simply as follows.

```
/* A AND Z */
A_AND_Z=BAND(RANK('A'),RANK('Z'));

/* A OR Z */
A_OR_Z=BOR(RANK('A'),RANK('Z'));

/* A XOR Z */
A_XOR_Z=BXOR(RANK('A'),RANK('Z'));
```

To show the character representation of A_AND_Z, A_OR_Z and A_XOR_Z, we can use BYTE function in SAS:

```
/* RETURNS '@' */
C_AND=BYTE(A_AND_Z);

/* RETURNS '[' */
C_OR=BYTE(A_OR_Z);

/* RETURNS A FUNNY CHARACTER RELATED TO
ASCII CODE 27 */
C_XOR=BYTE(A_XOR_Z,);
```

As you can see, the new bitwise functions are much better in coding efficiency.

The choices of the keys or passwords in SAS coding are numerous. Authors would like to present 4 options of the keys based on their own experiences.

The first option is to simply use ranged random numbers as the keys to scramble credit card number, as shown in following SAS codes:

Box 1: Random Number as the Key

```
%LET PASSWD=54321;
...
DO I=1 TO 16;
C=RANK(SUBSTR(CARD,I,1));
P=INT(UNIFORM("&PASSWD")*255);
Y=INT(UNIFORM("&PASSWD")*127)+128;
X=BXOR(C,P);
Z=BXOR(X,Y);
SUBSTR(CARD,I,1)=BYTE(Z);
END;
```

In this SAS program, both variables P and Y are keys from random numbers. The variable Z represents scrambled value of CARD. The macro variable &PASSWD serves as seed in UNIFORM function. We recommend to select the seed larger than zero, in order to keep the same random number stream for both encryption and decryption. Because the same characters are unlikely to be scrambled by the same random number, the pattern of scrambled characters is very hard to identify. However, this key is sensitive to the change of record order of the file. If other party send back the same file with changed record order, the above program can not decrypt original value of CARD back..

To avoid of changing record order, the second option is to use hard coded password as shown below:

Box 2: Hard Coded Password as the Key

```
%LET PASSWD=MY_SECRETS;
...
DO I=1 TO 16;
K=1+MOD(I,LENGTH(LEFT("&PASSWD")));
X=BXOR(RANK(SUBSTR(CARD,I,1)),
   RANK(SUBSTR(LEFT("&PASSWD"),K,1)));
Y=BOR(254-BAND(254,I*K),BXOR(127,I*K));
Z=BXOR(X,Y);
SUBSTR(CARD,I,1)=BYTE(Z);
END;
```

In this program, &PASSWD is the hard coded key. The variable Y is another key based on the position of the CARD. Using this same program, the field can be encrypt or decrypt regardless the record order. However, if two records have the same character at the same column position, the scrambled characters will also be the same.

The third option is to use other fields of the same record to encrypt, as shown below.

Box 3: Other Field as the Key

```
INPUT @1 NAME $10. @11 CARD $CHAR16. @28 ZIP
5.;
DO I=1 TO 16;
X=RANK(SUBSTR(CARD,I,1));
K=MOD(I,LENGTH(NAME))+1;
P=RANK(SUBSTR(NAME,K,1));
CALL RANUNI(SUM(ZIP,54321), Y);
Y=INT(Y*127)+128;
Z=BXOR(X,P);
Z=BXOR(Z,Y);
SUBSTR(CARD,I,1)=BYTE(Z);
END;
```

Here both the variables NAME and ZIP are taken as keys to scramble the CARD. Because it is uncommon for two records to have the same pattern of the NAME, ZIP and CARD, the chance of repeating encrypted character from the same original character is very small. In addition, the decryption using the above SAS codes is not affected by any change of the record order. However, it is affected by the change in key variables, either their values or location.

The last option is just any combination of previous three options proposed.

Having demonstrated all these, the next section shows a complete sample SAS program that apply both hard coded password and other field as the key.

Sample SAS Codes and Results

The following SAS codes in Box 4 reads in a text file named 'TEXT.DAT' shown in Box 5, scrambles the file with password defined in a macro variable, and with another field called ZIP, then output the scrambled file named 'ENCODED.DAT' displayed in Box 6. Please note that you can also change formulas for assigning the password, and for assigning value of Y(i) in this program to create variety. This same program can also be used to decrypt the scrambled file as long as the password is the same. This program was executed in the UNIX environment. However, it can be used in the MVS/TSO, VM/CMS, MS/DOS etc.. The user should be aware of the difference between ASCII and EBCDIC though.

Box 4: Sample SAS Codes for the Keyed Encryption or Decryption

```
FILENAME IN 'TEXT.DAT';
FILENAME OUT 'ENCODED.DAT';

%LET PASSWD=MY_SECRETS;

DATA P;
INFILE IN;
INPUT  @11 CARD $CHAR16. @28 ZIP 5.;
DO I=1 TO 16;
K=1+MOD(I,LENGTH(LEFT("&PASSWD")));
X=BXOR(RANK(SUBSTR(CARD,I,1)),RANK(
    SUBSTR(LEFT("&PASSWD"),K,1)));
CALL RANUNI(SUM(ZIP,54321), Y);
Y=INT(Y*127)+128;
Z=BXOR(X,Y);
SUBSTR(CARD,I,1)=BYTE(Z);
END;
FILE OUT;
PUT _INFILE_ @11 CARD $CHAR16.;
RUN;
```

Box 5: 'TEST.DAT', the File to Be Scrambled

```
LIU       4138262726507865 388647667
ZHUANG    4148406071033483 218334264
SAIVE     4176527885516672 144016577
WALDNER   4186131877231482 763236878
PERSON    4108403267711763 444513513
KARLSON   5815886015563054 341028115
CHEN      5261605313831506 768240486
PRAMORIC  5686027022044006 387230016
ROSKOPHA  5337883151814075 485157245
GATIANN   5040804235307386 534781530
```

Box 6: 'ENCODED.DAT', the Scrambled File

```
LIU       °³½ ¬¹ª¾¼¦±²¹ ¨º 388647667
ZHUANG    ¼¿¶¬¦³¢µµ-¸½± ª° 218334264
SAIVE     “•š•ˆ•Œ’•†’•›•Š• 144016577
WALDNER   òñôìíþëóûåôóýîäÿ 763236878
PERSON    âáìòøíùéêõááíýúî 444513513
KARLSON   ”Ÿšˆƒ’‹œš•”‘˜••• 341028115
CHEN      ˜™‘••–„“–Š•˜–„‡• 768240486
PRAMORIC  ¿º¸  ³¡·²¬º¸´¦ · 387230016
ROSKOPHA  ŠŠ†”•Œ•ƒ•š‡ˆ•“’• 485157245
GATIANN   •Œ„–˜•’…ƒ›‰Œ‡•˜‡ 534781530
```

Summary

We discussed some concepts of keyed encrytion to data fields. We also compared old way to new way of encrypting data fields in SAS. The SAS new bitwise function certainly make the encryption much easier. We then presented 4 options of seleting keys. Finally, a sample SAS program and its result were demonstrated.

References

SAS Institute Inc. (1990), SAS Language, Reference Version 6, First Edition, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1991) SAS Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07, Cary, NC: SAS Institute Inc.

Annette I. Ladan, “Effective Data Encryption Algorithms Using SAS® System", Proceedinds of the 20th Annual SUGI Conference, pp451-452.

Robert Alonso, Quick C DOS Utilities, John Wiley & Son, Inc, 1988, pp92-92.

Author Contact

Sheng Luo
Providian Corporation
20 Moores Road
Frazer, PA 19355
Phone: (610)648-5065
Email: sheng_luo@providian.com

Xinsheng Lin
IMS America LTD.
600 W. Germantown Pike
Plymouth Meeting, PA 19462-0905
Phone: (610)832-5532
Email: xinshl@imsint.com