Effective Data Encryption Algorithms using the SAS® System

Annette I. Ladan, Centers for Disease Control and Prevention

ABSTRACT

It is desirable to disguise sensitive data, such as name and social security number from scrutiny or unauthorized access when using a shared SAS data set. A simple encryption/decryption algorithm to scramble key data solves this data security issue. Many SAS System character functions are available to support unique, simple, and keyless data encryption algorithms.

The cryptographic ciphers (substitution/translation schemes) presented for SAS character data are based on the monoalphabetic, polyalphabetic, and transposition approach. SAS-based encryption algorithms using each of these approaches and the precautions for devising such schemes will be described.

INTRODUCTION

A knowledgeable user can generate a proc contents and have access to variable descriptions on any unprotected SAS data set. Once the file is accessed, a user can expose, distribute, and/or manipulate sensitive data. Passwords, DOS commands, and various SAS System options and procedures can limit unauthorized access to a SAS data set. The use of encryption algorithms for sensitive data variables within the SAS data set provides additional assurance against unauthorized tampering especially when the data set is shared.

The many SAS System functions inherent in Base SAS software make it possible to develop such algorithms. Using these functions sensitive data are scrambled which makes the data set more secure.

SAS SYSTEM FUNCTIONS

The data encryption algorithms presented use many Base SAS System character functions. A brief overview of each function is listed.

COLLATE	generates a collating sequence string
LEFT	removes leading spaces and shifts text to the left margin
LENGTH	returns a numeric value of a text string
REVERSE	swaps the order of a character expression
SCAN	searches for words and gives the result specified in the input argument
SUBSTR	extracts a segment from a text string
TRANSLATE	substitutes characters with other characters
TRIM	removes trailing spaces

DATA ENCRYPTION ALGORITHMS

Encryption/decryption processes are either key (public and/or private key assignments) or keyless (secretly protected algorithms). The United States government's Data Encryption Standard (DES) is a key-based system. Cryptographic ciphers are non-keyed systems. The monoalphabetic, polyalphabetic, and transposition ciphers are the most basic non-keyed cryptographic systems. "The monoalphabetic cipher substitutes every letter in the data with a letter that is three (sic) times higher. The polyalphabetic cipher divides data into groups of letters and then shifts the letters of each group. The transposition cipher changes the order in which letters appear but not the letters themselves."[1] Data encryption algorithms using the monoalphabetic, polyalphabetic, and transposition ciphers are illustrated below.

```
*Sample PC SAS program. Outputs follow each data step. ;
DATA NAME;
 INPUT NAME $ 1-25;
STRING = COLLATE(65,90);   *Uppercase letters, A-Z;
HIGH2 = SUBSTR(STRING,2);  *Letters B-Z;
HIGH3 = SUBSTR(STRING,3);  *Letters C-Z;
HIGH4 = SUBSTR(STRING,4);  *Letters D-Z;
 CARDS;
JOHN DOE
;
RUN;
```

Example 1. Use the monoalphabetic substitution cipher to substitute every letter in the name with a letter that is three times higher.
Functions COLLATE, SUBSTR, TRANSLATE

```
DATA MONO;
SET NAME;
ENCRYPT = TRANSLATE(NAME,HIGH4,STRING);
DECRYPT = TRANSLATE(ENCRYPT,STRING,HIGH4);
PROC PRINT DATA=MONO;
VAR NAME ENCRYPT DECRYPT;
TITLE 'Illustration of the Monoalphabetic Cipher';
RUN;
```

Illustration of the Monoalphabetic Cipher

NAME	ENCRYPT	DECRYPT
JOHN DOE	MRKQ GRH	JOHN DOE

Example 2. Use the polyalphabetic substitution cipher to divide the first and last name into groups of letters and then shift the letters of each group.
Functions SCAN, TRIM, LENGTH

```
DATA POLY;
SET NAME;
LENGTH ENCRYPTF ENCRYPTL DECRYPTF DECRYPTL
FIRST LAST $ 10;
LENGTH ENCRYPT DECRYPT $ 25; *Set max name limit;
FIRST = TRIM(SCAN(NAME,1));
LAST = TRIM(SCAN,NAME,2));

DO I = 1 TO LENGTH(FIRST);*Encrypts/decrypts first name;
SUBSTR(ENCRYPTF,I,1) =
TRANSLATE(SUBSTR(FIRST,I,1),HIGH2,STRING);
SUBSTR(DECRYPTF,I,1) =
TRANSLATE(SUBSTR(ENCRYPTF,I,1),STRING,HIGH2);
END;

DO I = 1 TO LENGTH(LAST); *Encrypts/decrypts last name;
SUBSTR(ENCRYPTL,I,1) =
TRANSLATE(SUBSTR(LAST,I,1),HIGH3,STRING);
SUBSTR(DECRYPTL,I,1) =
TRANSLATE(SUBSTR(ENCRYPTL,I,1),STRING,HIGH3);
END;

DO I = 1 TO LENGTH(NAME); *Concatenates first/last name;
SUBSTR(ENCRYPT,I,25) =
((SUBSTR(ENCRYPTF,I,10))||(SUBSTR(ENCRYPTL,I,10)));
SUBSTR(DECRYPT,I,25) =
((SUBSTR(DECRYPTF,I,10))||(SUBSTR(DECRYPTL,I,10)));
END;
PROC PRINT DATA = POLY;
VAR NAME ENCRYPT DECRYPT;
TITLE 'Illustration of the Polyalphabetic Cipher';
RUN;
```

Illustration of the Polyalphabetic Cipher

NAME	ENCRYPT	DECRYPT
JOHN DOE	KPIO FQG	JOHN DOE

Example 3. Use the transposition cipher to change the order in which letters appear but not the letters themselves.
Function REVERSE, LEFT

```
DATA TRANS;
SET NAME;
LENGTH ENCRYPT DECRYPT;
ENCRYPT = LEFT(REVERSE(NAME));
DECRYPT = LEFT(REVERSE(ENCRYPT));
PROC PRINT DATA=TRANS;
VAR NAME ENCRYPT DECRYPT;
TITLE 'Illustration of the Transposition Cipher';
RUN;
```

Illustration of the Transposition Cipher

NAME	ENCRYPT	DECRYPT
JOHN DOE	EOD NHOJ	JOHN DOE

PRECAUTIONS

As with any new technique, it is important to test the encryption algorithm on a dummy SAS data set. Users should also take these precautions: (1) make sure data can be decrypted once encrypted; (2) do not create faulty or undecipherable encryption algorithms and keep the algorithm simple; (3) know the contents of the encrypted variable and keep record layouts handy; and (4) secure the SAS code as well as the encryption algorithm.

CONCLUSION

The monoalphabetic, polyalphabetic, and transposition ciphers are effective means for encrypting data only if the algorithm remains a secret. These data encryption algorithms will protect sensitive character data variables on a shared SAS data set.

REFERENCE

'Prosise, Jeff (1994), "How to Keep it a Secret," PC Magazine, pp. 315-318.

ACKNOWLEDGMENTS

Thanks to Van Munn for his technical review and to Sharon Dickerson, MPA for her editorial comments.