Paper 108-2010

# Data Masking with Classical Ciphers

Murphy Choy, University College Dublin

## ABSTRACT

Data masking and protection is an essential part of data processing. With easy access to data, encryption of sensitive information is now a critical part of the data process especially with respect to the new data regulations in place for many countries. While there are no explicit data encryption functions in SAS® , many classical encryption ciphers can be coded in SAS® and applied on the data. In this paper, we will present 3 classical ciphers macro, Caesar's cipher, Alberti cipher and BIFID cipher.

## INTRODUCTION

Data privacy has been a major problem for many organizations in the recent years. To prevent possible data thefts, many database companies have started developing data encryption algorithms to ensure that data will be kept private at all time and cannot be abused easily. However, this functionality is absent from base SAS® . To allow SAS® users and administrators to be able to mask their data, three SAS® macros have been developed based on the classical ciphers with both encryption and decryption functionality built into them.

## CAESAR'S CIPHER

Caesar's cipher is one of the earliest known ciphers that is being used for encryption purposes. As its name suggested, it is supposedly to be created and used by the great Roman general, Julius Caesar. The fundamental workings of the cipher is very simple and it is basically a shift in the in the alphabet order and substituting the values in the original string with the values in the new string. The operation can be expressed in a mathematical form which is modulus function.

The encryption function is expressed below as

$$E_n(x) = (x + n) \mod 26.$$

The decryption function is expressed below as

$$D_n(x) = (x - n) \mod 26.$$

Due to the nature of this shift, it is easily broken by a simple shift of the alphabet series. At the same time, given that the maximum shift is 26, it is possible to use a brute force approach to break through the cipher. Thus this cipher provides very little encryption protection.

## IMPLEMENTATION OF THE CAESAR'S CIPHER IN SAS®

```
/*************************************************************************
-------------------------------------------------------------------------
CAESAR'S CIPHER
-------------------------------------------------------------------------
THIS IS A SIMPLE IMPLEMENTATION OF THE CAESAR'S CIPHER WHICH IS THE SIMPLEST
CIPHER THAT CAN BE USED FOR ENCRYPTION PURPOSES.
-------------------------------------------------------------------------
INPUT                          DESCRIPTION
-------------------------------------------------------------------------
INPUT                          INPUT DATA
VAR                            VARIABLE TO BE ENCRYPTED
SHIFT                          SHIFTING DEGREE
MODE                           ENCRYPTION/DECRYPTION (ENCRYPT/DECRYPT)
```

```
-------------------------------------------------------------------------------
*******************************************************************************/

%LET INPUT = ;
%LET VAR = ;
%LET SHIFT = ;
%LET MODE = ;

/*******************************************************************************/

/*******************************************************************************
MAIN MACRO
*******************************************************************************/

%MACRO CAESAR_CIPHER();

        /*******************************************************
        GENERATION OF THE CIPHER THROUGH CHARACTER SHIFTS
        *******************************************************/

        DATA _NULL_;

                /*ALPHABETICAL STRING: 26 CHARACTERS*/
                ORIGINAL = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

                /*SHIFTING THE STRING BY THE NUMBER OF CHARACTER*/
                CIPHER = COMPRESS(SUBSTR(ORIGINAL,&SHIFT+1)
                        %DO I = 1 %TO &SHIFT;
                                ||SUBSTR(ORIGINAL,&I,1)
                        %END;
                        );

                /*ASSIGNING A MACRO VARIABLE FOR SUBSEQUENT USE*/
                CALL SYMPUT('ORIGINAL',COMPRESS(ORIGINAL));
                CALL SYMPUT('CIPHER',COMPRESS(CIPHER));

        RUN;

        /*******************************************************
        ENCRYPTION OR DECRYPTION OF DATA
        *******************************************************/

        DATA &INPUT;
        SET &INPUT;

                /*ENCRYPTION*/
                %IF &MODE = "ENCRYPT" %THEN %DO;
                        ENCRYPTED_&VAR = TRANSLATE(&VAR,"&ORIGINAL","&CIPHER");
                %END;
                /*DECRYPTION*/
                %ELSE %IF &MODE = "DECRYPT" %THEN %DO;
                        DECRYPTED_&VAR = TRANSLATE(ENCRYPTED_&VAR,"&CIPHER","&ORIGINAL");
                %END;
                /*NO ACTION*/
                %ELSE %DO;
                        &VAR = &VAR;
                %END;
        RUN;

%MEND;

/*******************************************************************************/

/*Macro calling*/
%CAESAR_CIPHER();
```

## Alberti's cipher

Alberti's cipher is a cipher first described by Leon Battista Alberti in the mid 15th century. The method uses a wheel with 2 circles with the inner circle marking the original alphanumeric list to be substituted and the outer circle being the one that indicates the value that is to be used for substitution. After the initial shift, the substitution wheel changes for every Nth words by a shift of K. Due to this ever changing shifts, it is a relatively stronger encryption compared to our Caesar's cipher.

To implement the Alberti's cipher, we have to develop an ever shifting substitution string which is then always applied to the data.

## IMPLEMENTATION OF THE ALBERTI'S CIPHER IN SAS®

```
/*************************************************************************
-------------------------------------------------------------------------
ALBERTI CIPHER
-------------------------------------------------------------------------
ALBERTI CIPHER IS ONE OF THE OLDEST CIPHER WHICH IS USED TO ENCRYPT INFORMATION
AND IT IS CONSIDERED TO BE ONE OF THE MOST RELIABLE ENCRYPTION METHOD IN ITS
TIME. THIS MACRO IS A SIMPLE IMPLEMENTATION OF THE ALBERTI CIPHER.
-------------------------------------------------------------------------
INPUT                          DESCRIPTION
-------------------------------------------------------------------------
INPUT                          INPUT DATA
SHIFT_1                        PRELIMINARY STARTING POSITION
SHIFT_2                        NEXT POSITION TO SHIFT
SHIFT_3                        SUBSEQUENT SHIFTS
VAR                            VARIABLE TO BE ENCRYPTED
MODE                           ENCRYPT/DECRYPT
-------------------------------------------------------------------------
*************************************************************************/

%LET INPUT = ;
%LET SHIFT_1 = ;
%LET SHIFT_2 = ;
%LET SHIFT_3 = ;
%LET VAR = ;
%LET MODE = ;

/*************************************************************************/

/*************************************************************************
MAIN MACRO
*************************************************************************/

%MACRO ALBERTI_CIPHER();

        DATA &INPUT;
        SET &INPUT;

        /*RETAINING CIPHER TEXT*/
        RETAIN CIPHER _CHAR_ ;
        RETAIN COUNT;

        /*ENCRYPTION SECTION*/
        %IF &MODE = ENCRYPT %THEN %DO;
                /*DECLARING THE ORIGINAL STRING*/
                ORIGINAL = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

                /*FIRST TIME INITIALIZATION*/
                IF _N_ = 1 THEN DO;

                        CIPHER = COMPRESS(SUBSTR(ORIGINAL,&SHIFT_1+1)
                        %DO I = 1 %TO &SHIFT_1;
                                ||SUBSTR(ORIGINAL,&I,1)
                        %END;
                        );

                        ENCRYPTED_&VAR = TRANSLATE(&VAR,ORIGINAL,CIPHER);
                        COUNT = 1;
                END;
```

```
                /*EVERY NTH TIME REINITIALIZATION*/
                ELSE IF COUNT = &SHIFT_2 THEN DO;

                        CIPHER = COMPRESS(SUBSTR(CIPHER,&SHIFT_3+1)
                        %DO I = 1 %TO &SHIFT_3;
                                ||SUBSTR(CIPHER,&I,1)
                        %END;
                        );

                        ENCRYPTED_&VAR = TRANSLATE(&VAR,ORIGINAL,CIPHER);
                        COUNT = 1;
                END;

                /*ENCRYPTION BETWEEN EVERY REINITIALIZATION*/
                ELSE DO;
                        ENCRYPTED_&VAR = TRANSLATE(&VAR,ORIGINAL,CIPHER);
                        COUNT = COUNT + 1;
                END;

        %END;
        %ELSE %IF &MODE = DECRYPT %THEN %DO;

                /*DECLARING THE ORIGINAL STRING*/
                ORIGINAL = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

                /*FIRST TIME INITIALIZATION*/
                IF _N_ = 1 THEN DO;

                        CIPHER = COMPRESS(SUBSTR(ORIGINAL,&SHIFT_1+1)
                        %DO I = 1 %TO &SHIFT_1;
                                ||SUBSTR(ORIGINAL,&I,1)
                        %END;
                        );

                        DECRYPTED_&VAR = TRANSLATE(&VAR,CIPHER,ORIGINAL);
                        COUNT = 1;
                END;

                /*EVERY NTH TIME REINITIALIZATION*/
                ELSE IF COUNT >= &SHIFT_2 THEN DO;

                        CIPHER = COMPRESS(SUBSTR(CIPHER,&SHIFT_3+1)
                        %DO I = 1 %TO &SHIFT_3;
                                ||SUBSTR(CIPHER,&I,1)
                        %END;
                        );

                        DECRYPTED_&VAR = TRANSLATE(&VAR,CIPHER,ORIGINAL);
                        COUNT = 1;
                END;

                /*DECRYPTION BETWEEN EVERY REINITIALIZATION*/
                ELSE DO;
                        DECRYPTED_&VAR = TRANSLATE(&VAR,CIPHER,ORIGINAL);
                        COUNT = COUNT + 1;
                END;
        %END;

                DROP COUNT CIPHER ORIGINAL;

        RUN;


%MEND;

/*************************************************************************
MACRO CALLING
*************************************************************************/

%ALBERTI_CIPHER();
```

## BIFID CIPHER

The BIFID cipher is a very complex cipher incorporating elements from polybius square with transposition and fractionation to achieve diffusion. Compared to the previous 2 examples, it is a far more complex algorithm in terms of amount of manipulation of characters. The first fundamental difference lies in the presence of 2 string substitution which is then remade into a single string with the same values set within. It is also considered to be quite a strong form of encryption.

## IMPLEMENTATION OF THE BIFID CIPHER IN SAS®

```
/*************************************************************************
-------------------------------------------------------------------------
BIFID CIPHER
-------------------------------------------------------------------------
THIS MACRO IS A SIMPLE IMPLEMENTATION OF THE BIFID CIPHER
-------------------------------------------------------------------------
INPUT                  DESCRIPTION
-------------------------------------------------------------------------
INPUT                  INPUT DATA SET
VAR                    VARIABLE TO BE USED
SEED                   THE KEY
MODE                   ENCRYPT/DECRYPT
-------------------------------------------------------------------------
*************************************************************************/

%LET INPUT = ;
%LET VAR = ;
%LET SEED = ;
%LET MODE = ;


/*************************************************************************/

/************************************************************************
MAIN MACRO
*************************************************************************/

%MACRO BIFID_CIPHER();

/************************************************************************
POLYBIUS CREATION SECTION
*************************************************************************/

/*INITIALIZATION OF THE POLYBIUS TABLE*/

DATA POLYBIUS_TABLE;
ORIGINAL = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ123456789.';
LENGTH_OF_STRING = LENGTH(ORIGINAL);

DO I = 1 TO LENGTH_OF_STRING;
        /*SELECTING EACH LATTER AND OUTPUTTING THEM*/
        WORD = SUBSTR(ORIGINAL,I,1);
        OUTPUT;
END;

DROP ORIGINAL LENGTH_OF_STRING I;

RUN;

/*CREATING RANDOMNESS TO PRODUCE SQUARES*/

DATA POLYBIUS_TABLE;
SET POLYBIUS_TABLE;

        /*PRODUCING THE RANDOMNESS*/
        CHANCE = RANUNI(&SEED);
```

```
RUN;

/*SORTING THE TABLE SIMULATE RANDOMNESS IN TABLE*/

PROC SORT DATA = POLYBIUS_TABLE;
BY CHANCE;
RUN;

/*SETTING THE POSITION OF THE POLYBIUS TABLE*/

DATA POLYBIUS_TABLE;
SET POLYBIUS_TABLE;

RETAIN X Y COUNT;

/*FIRST INITIALIZATION*/
IF _N_ = 1 THEN DO;
        X = 1;
        Y = 1;
        COUNT = 1;
END;
/*REPETITION FOR THE SQUARE BEHAVIOR*/
ELSE IF COUNT = 6 THEN DO;
        X = 1;
        Y = SUM(Y,1);
        COUNT = 1;
END;
/*INTERMEDIATE CALCULATIONS*/
ELSE DO;
        X = SUM(X,1);
        COUNT = SUM(COUNT,1);
END;

RUN;

/*************************************************************************
VALUE ASSIGNING SECTION
*************************************************************************/

/*CIPHER CREATION SECTION*/

DATA _NULL;
SET POLYBIUS_TABLE;

/*LENGTH DECLARATIONS*/

LENGTH CIPHER1 $45 CIPHER2 $45 KEY $45;

/*KEY RETENTION FOR BOTH CIPHER AND DECIPHER*/
RETAIN CIPHER1;
RETAIN CIPHER2;
RETAIN KEY;

/*ENCRYPTION KEY AND CIPHERS*/
KEY = COMPRESS(KEY||WORD);
CIPHER1 = COMPRESS(CIPHER1||X);
CIPHER2 = COMPRESS(CIPHER2||Y);

/*ASSIGNMENT OF KEY AND CIPHER AS MACRO VALUES*/
CALL SYMPUT('CIPHER1',COMPRESS(CIPHER1));
CALL SYMPUT('CIPHER2',COMPRESS(CIPHER2));
CALL SYMPUT('KEY',COMPRESS(KEY));

/*SUBSTITUTION KEY SECTION*/

SUB_KEY = COMPRESS(X||Y);

CALL SYMPUT('SUBKEY'||TRIM(LEFT(_N_)),SUB_KEY);
CALL SYMPUT('LETTER'||TRIM(LEFT(_N_)),WORD);

PUT SUB_KEY = KEY = CIPHER1 = CIPHER2 =;
```

```
RUN;

/*************************************************************************
LENGTH CHECK
*************************************************************************/

PROC SQL NOPRINT;

        /*ENSURING THAT THE TOTAL LENGTH WILL BE AVAILABLE FOR USE*/
        SELECT 2*MAX(LENGTH(&VAR)) INTO :MAX_LENGTH FROM &INPUT;

QUIT;

/*************************************************************************
ENCRYPTION/DECRYPTION SECTION
*************************************************************************/

DATA &INPUT;
SET &INPUT;

/*ENCRYPTION SECTION*/
%IF &MODE = ENCRYPT %THEN %DO;

/*DECLARATION OF THE LENGTH OF THE ENCRYPTED FIELD*/
LENGTH ENCRYPTED_&VAR $ &MAX_LENGTH;

        /*REMOVAL OF SPACES WITH DECIMALS*/
        &VAR = SUBSTR(TRANSLATE(&VAR,'.',' '),1,LENGTH(&VAR));

        /*CREATING CIPHER KEYS*/
        CIPHER1_&VAR = TRANSLATE(&VAR,"&CIPHER1","&KEY");
        CIPHER2_&VAR = TRANSLATE(&VAR,"&CIPHER2","&KEY");

        /*COMBINED CIPHER KEY*/
        CIPHER_&VAR = COMPRESS(CIPHER1_&VAR||CIPHER2_&VAR);

        /*SUBSTITUTION OF KEYS*/
        DO I = 1 TO LENGTH(CIPHER_&VAR)-1 BY 2;

                /*INITIALIZATION*/
                IF I = 1 THEN ENCRYPTED_&VAR = '';

                /*SUBSTITUTION STEPS*/
                IF SUBSTR(CIPHER_&VAR,I,2) = COMPRESS("&SUBKEY1") THEN ENCRYPTED_&VAR =
COMPRESS(ENCRYPTED_&VAR||"&LETTER1");

                %DO J = 1 %TO 36;

                        ELSE IF SUBSTR(CIPHER_&VAR,I,2) = COMPRESS("&&SUBKEY&J") THEN
ENCRYPTED_&VAR = COMPRESS(ENCRYPTED_&VAR||"&&LETTER&J");

                %END;

        END;

        /*REMOVAL OF UNNECESSARY COLUMNS*/
        DROP I CIPHER1_&VAR CIPHER2_&VAR CIPHER_&VAR;

%END;
%ELSE %IF &MODE = DECRYPT %THEN %DO;

/*DECLARATION OF THE DECRYPTION FIELD*/
LENGTH DECIPHER_KEY $ &MAX_LENGTH DECIPHER_KEY1 $ &MAX_LENGTH DECRYPTED_&VAR $
&MAX_LENGTH;

        /*CREATING THE DECIPHER KEY*/
        DECIPHER1_&VAR = TRANSLATE(&VAR,"&CIPHER1","&KEY");
        DECIPHER2_&VAR = TRANSLATE(&VAR,"&CIPHER2","&KEY");

        /*CREATING COMBINED DECIPHER KEY*/
```

```
        DO I = 1 TO LENGTH(&VAR);

                /*INITIALIZATION*/
                IF I = 1 THEN DECRYPTED_&VAR = '';

                DECIPHER_KEY = COMPRESS(DECIPHER_KEY
                                        ||SUBSTR(DECIPHER1_&VAR,I,1)
                                        ||SUBSTR(DECIPHER2_&VAR,I,1));

        END;

        /*CREATING THE ORIGINAL CIPHER KEY*/
        DO I = 1 TO LENGTH(&VAR);

                /*INITIALIZATION*/
                IF I = 1 THEN DECRYPTED_&VAR = '';

                DECIPHER_KEY1 = COMPRESS(DECIPHER_KEY1
                                        ||SUBSTR(DECIPHER_KEY,I,1)
                                        ||SUBSTR(DECIPHER_KEY,LENGTH(&VAR)+I,1));

        END;

        /*SUBSTITUTION OF THE LETTERS*/
        DO I = 1 TO LENGTH(DECIPHER_KEY1)-1 BY 2;

                /*INITIALIZATION*/
                IF I = 1 THEN DECRYPTED_&VAR = '';

                /*SUBSTITUTION STEPS*/
                IF SUBSTR(DECIPHER_KEY1,I,2) = COMPRESS("&SUBKEY1") THEN DECRYPTED_&VAR =
COMPRESS(DECRYPTED_&VAR||"&LETTER1");

                %DO J = 1 %TO 36;

                        ELSE IF SUBSTR(DECIPHER_KEY1,I,2) = COMPRESS("&&SUBKEY&J") THEN
DECRYPTED_&VAR = COMPRESS(DECRYPTED_&VAR||"&&LETTER&J");

                %END;

        END;

        /*RETRIVAL OF SPACES WITH DECIMALS*/
        DECRYPTED_&VAR = TRANSLATE(DECRYPTED_&VAR,' ','.');

        /*REMOVAL OF UNNECESSARY COLUMNS*/
        DROP I DECIPHER1_&VAR DECIPHER2_&VAR DECIPHER_KEY DECIPHER_KEY1;

%END;

/*************************************************************************/

RUN;

/*************************************************************************/

%MEND;

/*************************************************************************/

/*************************************************************************
MACRO CALLING
*************************************************************************/

%BIFID_CIPHER();
```

## CONCLUSION

Using SAS® to encrypt and mask your data is a simple task using TRANSLATE and SUBSTR functions.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Murphy Choy
E-mail: goladin@gmail.com