

Using CALL RANUNI for Encrypting Identifier Fields

H. Ian Whitlock, Westat Inc.

Introduction

Survey work often involves encrypting an identifier field when a file is produced for public use. How should identifier fields be encrypted in SAS®? The encryption should involve more than a simple substitution of characters. It should be possible to encrypt new data without interfering with previously encrypted data.

Without any claim to expertise in cryptography, I intuitively believe that a secure SAS encrypting method can be built on RANUNI.

This paper demonstrates a possible solution. The method uses the simple fixed place substitution of character for character, but the substitution table changes with each ID depending on the values in certain positions.

For simplicity, the code given assumes positions 7 and 8 contain information about the substitution table. This means the identifying field must be at least 8 bytes, and as written, the code requires at least 9 bytes.

The Method

Three macros generating partial DATA step code are employed:

%ENCRYPT -

Defines a random substitution table based on a given seed value.

%ENCODE -

Constructs the seed corresponding to a value of the ID field, invokes ENCRYPT to get the substitution table, and uses it to encode the field.

%DECODE -

Constructs the seed corresponding to an encoded ID, invokes ENCRYPT to get the substitution table, and uses it to retrieve the original value.

Three temporary variables are created and dropped by these macros - __I, __X, and __TS.

All the parameters in these macros name DATA step variables. Each parameter has its own name as the default value.

To use the macros in a DATA step:

- 1) Provide a list of all relevant characters in a retained variable named by the parameter BASE.
- 2) Define an 8 or 9 digit integer key in a retained variable named by the parameter KEY. The algorithm is essentially a function of this value and the positions 7 and 8 in the ID.
- 3) Provide two character variables named by the parameters TARGET and CODE. These variables should be equal in length to the variable named by the parameter BASE. The CODE variable holds a fixed substitution table used for positions 7 & 8 which depends on the key value. The TARGET variable holds a temporary substitution table dependent on the key value and what is in positions 7 and 8 of the identifier.
- 4) Provide a seed variable named by the parameter SEED.
- 5) Define the SEED variable to be the key value and invoke ENCRYPT once to fill in the CODE variable.
- 6) To encode an identifier, call ENCODE, with ID naming the identifier. ENCODE invokes ENCRYPT to fill in the TARGET variable depending on positions 7 and 8 in the identifier.
- 7) To decode a coded identifier, call DECODE, with ID naming the identifier. ENCODE invokes ENCRYPT to fill in the TARGET variable depending on positions 7 and 8 in the coded identifier.

The Macros

```
%macro encrypt ( seed = seed , base = base ,
                target = target ) ;
    drop __i __x __ts ;
    &target = '' ;
    __i = 1 ;
    do until ( index ( &target , '' ) = 0 ) ;
        call ranuni ( &seed , __x ) ;
        __x = int ( __x * length ( &base ) ) + 1 ;
        if substr ( &target , __x , 1 ) = '' then
            do ;
                substr ( &target , __x , 1 ) = substr ( &base , __i , 1 ) ;
                __i + 1 ;
            end ;
    end ;
```

```

end;
%mend encrypt;

%macro encode ( base = base , target = target ,
               code = code , key = key , id = id );
    __x = index (&code, substr(&id,7,1)) * index (&code,
    substr(&id,8,1));
    __ts = length ( base );
    __ts = int ( (&key / (__ts * __ts)) * __x );
    %encrypt ( seed = __ts, base = &base, target = &target )
    do __i = 7, 8;
        substr ( &id, __i, 1 ) =
        substr ( &code, index(&base,substr(&id,__i,1)), 1 );
    end;
    do __i = 1 to 6, 9 to 13;
        substr ( &id, __i, 1 ) =
        substr ( &target, index(&base,substr(&id,__i,1)),
        1 );
    end;
%mend encode;

```

```

%macro decode ( base = base , target = target ,
               code = code , key = key , id = id );
    do __i = 7, 8;
        substr ( &id, __i, 1 ) =
        substr ( &base, index(&code,substr(&id,__i,1)), 1 );
    end;
    __x = index (&code, substr(&id,7,1)) * index (&code,
    substr(&id,8,1));
    __ts = length ( base );
    __ts = int ( (&key / (__ts * __ts)) * __x );
    %encrypt ( seed = __ts, base = &base, target = &target )
    do __i = 1 to 6, 9 to 13;
        substr (&id,__i,1) = substr ( &base,
        index(&target,substr(&id,__i,1)),1 );
    end;
%mend decode;

```

Test Code

```

data;
    retain base '0123456789abcdefghijklmnopqrstuvwxyz'
           key 123456789;
    length code target $ 36;
    seed = key;
    %encrypt ( target = code )
    put base = $char36. / code = $char36.;

    do j = 1 to 4;
        input subject1 $char13.;
        subject2 = subject1;
        %encode ( id = subject2 );
        subject3 = subject2;
    end;

```

```

%decode ( id = subject3 )
put subject1 = $char13. /
    subject2 = $char13. /
    subject3 = $char13.;

end;
stop;

cards;
s12345v67890x
abcdefghijklmnopqrstuvwxyz
s12345v67890x
abcdefghijklmnopqrstuvwxyz
run;

```

Enhancements

As presented, the key value is used to determine the code for positions 7 and 8, and it is also used to determine the seed values in the macros ENCODE and DECODE. There is no reason why there cannot be two distinct keys for the two purposes. No changes are necessary in the macros to implement the double key.

The Nth character of the encoded identifier corresponds to the Nth character of the original. The addition of a fixed permutation is possible.

The special role of positions 7 and 8 are hard coded into the macros. They could be made parameter values. Moreover, there is nothing sacred about using two positions and there is no need for them to be contiguous.

I have assumed the input identifier and the output encoded identifier are the same variable. Dropping this assumption gives more possible encryption algorithms. For example the encoded identifier could be longer than the original.

Conclusion

Without much trouble one can construct a relatively secure encryption algorithm in SAS using CALL RANUNI.

The author can be contacted by mail at

H. Ian Whitlock
Westat
1650 Research Boulevard
Rockville, MD 20850-3129

or by e-mail at

Whitloi1%westat@mcimail.com

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.