Paper 949-2017

Personally Identifiable Information Secured Transformation

Shuhua Liang, Zoe Bider, Kaiser Permanente Southern California

ABSTRACT

Organizations that create and store Personally Identifiable Information (PII) are often required to de-identify sensitive data to protect individuals' privacy. There are multiple methods that SAS can be used to de-identify PII depending on data types and encryption needs. The first method is data step encryption, which uses a built-in SAS function to password protect datasets. The second method is to apply a crosswalk mapping by linking a dataset with PII to a secured dataset that contains the PII and its corresponding surrogate, which is then used to replace the PII in the original dataset. The third method is in-SAS encryption which involves translating PII into an encrypted string using SAS functions; this could be a one-byte-to-one-byte, a one-byte-to-multiple-byte swap, or performing a translation on an external databases like Teradata and Oracle. This paper aims to discuss the advantages and disadvantages of these three methods, provide sample SAS codes, and finally, describe the corresponding methods to decrypt the encrypted data.

INTRODUCTION

With the proliferation of healthcare data from sources such as electronic medical records, Personally Identifiable Information (PII) has become more widely available. PII is defined as, "information that can be used to distinguish or trace an individual's identity, either alone or when combined with other personal or identifying information that is linked or associated with a specific individual, such as date and place of birth, mother's maiden name, etc." (U.S. General Services Administration, 2015). It is often necessary that PII is collected and stored for organizations such as hospitals and banks, but actions must be taken to ensure it is only viewed by those who are authorized to see it.

There are two main types of PII: direct and indirect identifiers. Direct PII is defined as data that can be directly linked to an individual's identity, for example, name, social security number, or medical record number, whereas indirect PII consists of identifiers that can indirectly be linked to an individual's identity, such as an extreme age, rare disease, etc. (Department of Labor, 2015). The sharing and utilization of data with PII is heavily regulated, creating a need for de-identification of this data when stored and shared. There are three main ways of protecting PII during storage and transmission of data that will be explored in this paper:

- Data step encryption
- Cross-walk: an overall translation method
- Translation: a one-to-one or one-to-many translation

These methods, either used individually or in combination, can prevent a PII data breach, which could lead to devastating consequences for both the individuals whose data are involved and the organizations holding the data.

The aim of this paper is to outline three common methods of de-identifying PII using SAS and compare the advantages and disadvantages of each method, discuss the methods to restore original data, and provide sample SAS codes.

DATA STEP ENCRYPTION

BACKGROUND

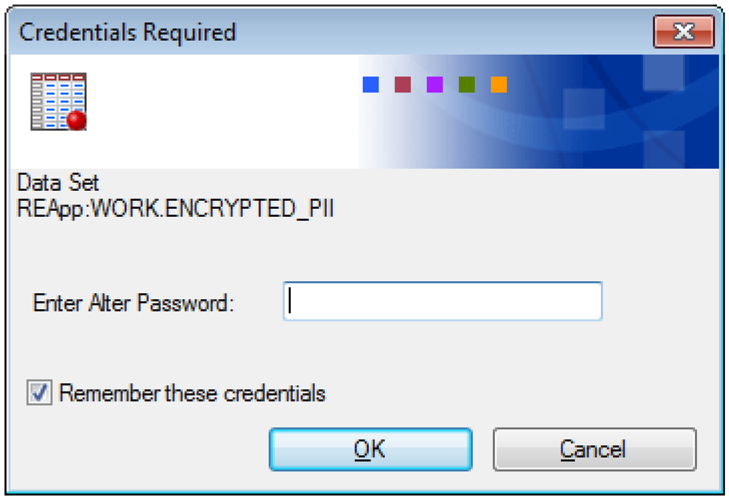
The simplest form of data encryption is to create a dataset that requires a password to open by storing a password in the SAS dataset. This encryption method has the advantage of being a simple built-in function that requires little coding. The disadvantages of this option is that it only provides a medium level of security, as it is vulnerable to brute-force hacking due to the finite number of possible password combinations. If this is a concern, a similar option is to include the Advanced Encryption Standard (AES) algorithm. The AES encryption provides a higher level of security by using a key/password value that is up to 64 characters long and is not stored in the dataset. This enhanced encryption is by SAS /SECUR software and available in SAS 9.4.

METHOD

In the DATA step, include the “encrypt” and “read” options. By default the encrypt option is set to “NO” unless user specifics the need for a password encryption. The read option allows the user to set a password for the dataset. The addition of the “alter” option prevents the data set from being replaced or deleted by requiring a password. In this example, a dataset entitled Encrypted_PII with two variables, OBS and PII are created, and encrypted with the password “SASGF”:

```
data Encrypted_PII (encrypt=yes read=SASGF alter=SASGF);
  input OBS $2. PII $11.;
  cards;
    01 123-45-5525
    02 643-54-0556
    03 325-46-7536
    04 732-12-1283
    05 908-66-1237
    06 242-53-7830
;
run;
```

Once the initial DATA step is run with the “encrypt” and “read” options, a pop-up window will display (Display 1). When the correct password is given, the dataset will be displayed, and if the “Remember these credentials” check box is selected, the data set will be unlocked for the remainder of the SAS session.


Credentials Required

Data Set
REApp:WORK.ENCRYPTED_PII

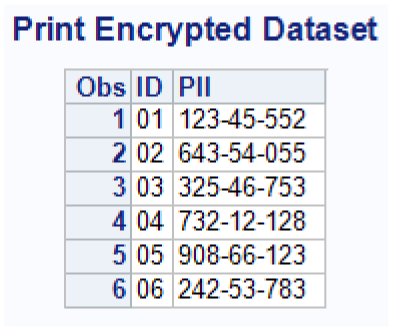
Enter Alter Password:

☑ Remember these credentials

Display 1: Pop-up window displayed

When calling the dataset using a SAS procedure such as the PRINT procedure, the “read=*password*” option must be used and the results are shown in Output 1:

```
proc print data=Encrypted PII (read=SASGF);
title 'Print Encrypted Dataset';
run;
title;
```


Print Encrypted Dataset

Obs	ID	PII
1	01	123-45-552
2	02	643-54-055
3	03	325-46-753
4	04	732-12-128
5	05	908-66-123
6	06	242-53-783

Output 1 – viewing printed encrypted dataset with password.

With the alter option protection, an error will prompt when we try to delete the dataset without credentials. To delete the dataset, the "alter" option with a valid password must be included.

```
proc datasets lib=work alter=SASGF;
  delete Encrypted_PII;
run;
```

CROSSWALK MAPPING

BACKGROUND

Crosswalk mapping is a method of encryption that is often useful when a dataset is used by multiple people, including those who need to see the PII and those who do not. In this case, the de-identified data can be shared, without sharing the components of the data with PII. The dataset creator creates a version of the dataset without PII that has a primary key that can be linked to the original dataset. This is ideal in situations where PII is used for dataset creation, but is not needed for data analysis. The downside is the lack of PII storage security for the dataset with PII. To address this issue, this encryption method can be combined with the DATA step encryption method described previously to password protect the original dataset.

METHOD

The first step in the crosswalk mapping method of encryption is to create two "librefs", one to store the PII data, and one to share the non-PII version of the data.

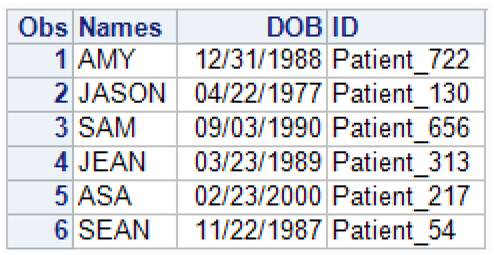
```
libname Secured 'Limited access directory that keeps PII data';
libname Share 'Directory that keeps the non-PII version of data for analyses';
```

Next, a variable with a random identifier will be added to the original PHI data. This random ID variable can be any length desired within system limit. In this example, a randomly generated number between 1 and 1000 is assigned to each record as a unique ID. The word "patient" is added as a prefix to the ID, which is optional.

```
/*Assign Random ID*/
%let seed = 54323;
data Secured.Crosswalk;
  set Original_PII;
array isAssigned[1:1000] _temporary_ (1000*0);
do while (1);
  newId = ceil(1000*ranuni(&seed));
  if isAssigned[newId] then continue;
  isAssigned[newId] = 1;
  leave;
```

```
end;

/*Convert randomly generated numbers into characters*/
ID = compress('Patient_' || newId);

keep Names ID DOB;
run;
```

The crosswalk dataset is produced, and contains both the original PII variable, and the new non-PII identifying variable (output 2).

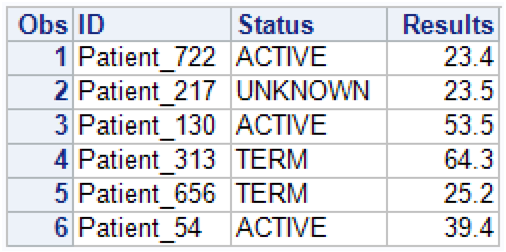


Obs	Names	DOB	ID
1	AMY	12/31/1988	Patient_722
2	JASON	04/22/1977	Patient_130
3	SAM	09/03/1990	Patient_656
4	JEAN	03/23/1989	Patient_313
5	ASA	02/23/2000	Patient_217
6	SEAN	11/22/1987	Patient_54

Output 2 – Dataset with PII and new non-PII identifier

Once the new random ID variable has been created, it is merged onto the original. In this PROC SQL step, only the non-PII ID variable is merged from the crosswalk dataset, and the PII identifying variable is not selected to be merged. The final dataset will include all the non-PII variables from the original dataset along with the new non-PII identifying variable (Output 3).

```
proc sql;
create table Share.Shared non PII as
select B.ID, A.Status, A.Results
from Original PII A left join Shared.Crosswalk B
on A.Names = B.Names and A.DOB = B.DOB;
quit;
```



Obs	ID	Status	Results
1	Patient_722	ACTIVE	23.4
2	Patient_217	UNKNOWN	23.5
3	Patient_130	ACTIVE	53.5
4	Patient_313	TERM	64.3
5	Patient_656	TERM	25.2
6	Patient_54	ACTIVE	39.4

Output 3 – Dataset with non-PII identifier that can be linked back to dataset with PII identifier.

TRANSLATION

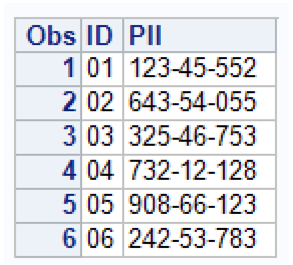
BACKGROUND

Translation, or keyed encryption, is another method that can be used to mask PII so that datasets can be shared without privacy breach concerns. SAS has built-in functions that can encrypt PII by quickly translating it into a set of meaningless characters, and conversely, decrypt it back to the original characters. This method of encryption is very similar to the crosswalk mapping method with the advantage of using built-in SAS functions, and does not require securing a dataset with the original PII. Instead, the program with the encryption key must be securely stored. This method has the disadvantage of the potential to have the encryption string being “cracked”. It must also be noted that codes cannot be repeated when creating the encryption, or it will not be decryptable. Ultimately, if there is a safe method of storing the encryption program, the one-byte-to-one-byte in SAS encryption and translation method is a fast and easy way to encrypt SAS variables with PII that can later be decrypted.

METHOD

Encryption using one-byte-to-one-byte translation can be completed in one SAS DATA step. In this example, a variable containing a PII variable will be encrypted (output 4). This variable has 11 characters, all of which will be encrypted. Since this is one-byte-to-one-byte, each of the original PII characters will be replaced with a single character to create an encrypted variable that has the same number of characters as the original. The translate functions used in tandem replace the PII with the random set of characters input by the user and the DO LOOP indicates that all 11 characters are to be encrypted. Alternatively, the "tranwrd" function can be used in place of the translate function. This process is completed by replacing each character one by one until all 11 characters of PII are replaced with their corresponding new characters.

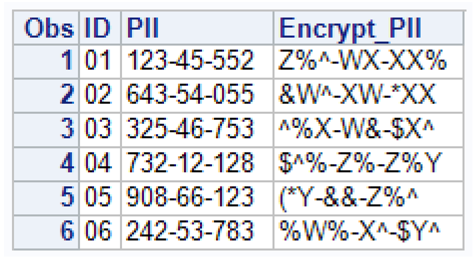
When creating your own encryption key, decide on a unique character for every unique character in the PII that needs to be encrypted. If the PII is an n digit number, then the encryption key needs to consist of n unique characters that can be letters, numbers or symbols. It is important to note that this method is limited to a PII variable that contains 95 characters, since that is the number of unique characters available to create the encryption key (output 5).



Obs	ID	PII
1	01	123-45-552
2	02	643-54-055
3	03	325-46-753
4	04	732-12-128
5	05	908-66-123
6	06	242-53-783

Output 4: original dataset with PII

```
/*Encryption*/
data Encrypt One To One;
  set Original One To One;
  length Encrypt PII $11.;
  do i = 1 to 11;
    encrypt PII=strip(encrypt_ssn)||translate(substr(ssn,i,1), '-Y*Z$%^WX&(',
'-8017234569');
  end;
  drop i ;
run;
```

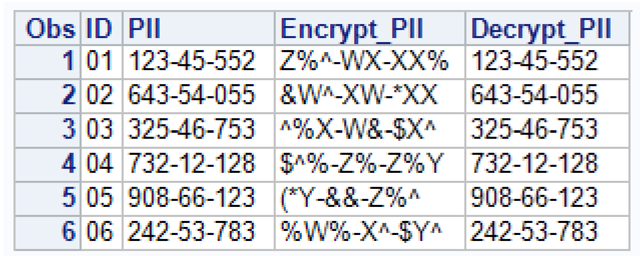


Obs	ID	PII	Encrypt_PII
1	01	123-45-552	Z%^-WX-XX%
2	02	643-54-055	&W^-XW-*XX
3	03	325-46-753	^%X-W&-$X^
4	04	732-12-128	$^%-Z%-Z%Y
5	05	908-66-123	(*Y-&&-Z%^
6	06	242-53-783	%W%-X^-$Y^

Output 5: dataset with original PII variable, and new encrypted variable created using one-byte to one-byte translation

The syntax for decryption is the same as encryption, except the encrypted variable is translated into the original PII variable (output 6).

```
/*Decryption*/
data Decrypt One To One;
  set Encrypt One To One;
  length Decrypt PII $11.;
  do j = 1 to 11;
       decrypt pii=strip(decrypt_ssn)||translate(substr(encrypt_ssn,j,1), '-
8017234569', '-Y*Z$%^WX&(');
  end;
  drop j;
run;
```

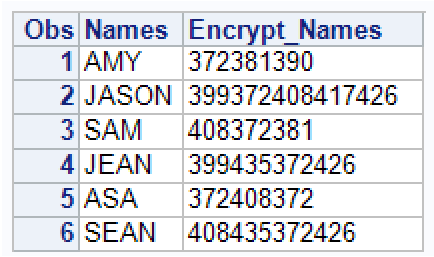


Obs	ID	PII	Encrypt_PII	Decrypt_PII
1	01	123-45-552	Z%^-WX-XX%	123-45-552
2	02	643-54-055	&W^-XW-*XX	643-54-055
3	03	325-46-753	^%X-W&-$X^	325-46-753
4	04	732-12-128	$^%-Z%-Z%Y	732-12-128
5	05	908-66-123	(*Y-&&-Z%^	908-66-123
6	06	242-53-783	%W%-X^-$Y^	242-53-783

Output 6: Original, encrypted and decrypted PII variable, example of one-byte to one-byte encryption

The same principles can be applied, but extending to one-to-multiple byte translations. In this example, each character of the PII is translated to three numbers, later converted to characters, based on the programmer's algorithm, with a pre-selected starting number ("k" in the following code) (Output 7).

```
%let multi = 3;
%let k = 372;
%let gap = 9;

/*Encryption*/
data Encrypt One To Multi;
  set Original_One_To_Multi;

  Encrypt Names = Names;
  i = input(&k., best.);
  do old = 'A','M','Y','J','S','O','N','E';
        new = put(i, &multi..);
        Encrypt Names = tranwrd(Encrypt_Names, old, new);
        i + input(&gap., best.);
  end;

  drop i new old;
run;
```



Obs	Names	Encrypt_Names
1	AMY	372381390
2	JASON	399372408417426
3	SAM	408372381
4	JEAN	399435372426
5	ASA	372408372
6	SEAN	408435372426

Output 7: dataset w ith original PII variable, and new encrypted variable created using one-byte to multiple-byte translation

As was the case with one-byte-to-one-byte translation, the syntax for decryption is the same as encryption, where the encrypted variable is translated into the original PII variable (Output 8).

```
/*Decryption*/
data Decrypt_One_To_Multi;
  set Encrypt_One_To_Multi;
  Decrypt_Names = Encrypt_Names;
  j = input(&k., best.);
  do new = 'A','M','Y','J','S','O','N','E';
        old = put(j, &multi..);
        Decrypt_Names = tranwrd(Decrypt_Names, old, new);
        j + input(&gap., best.);
  end;

  drop j new old;
run;
```

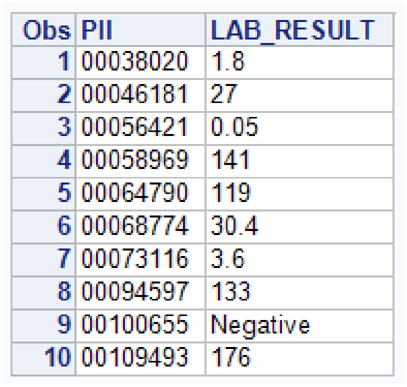
Obs	Names	Encrypt_Names	Decrypt_Names
1	AMY	372381390	AMY
2	JASON	399372408417426	JASON
3	SAM	408372381	SAM
4	JEAN	399435372426	JEAN
5	ASA	372408372	ASA
6	SEAN	408435372426	SEAN

Output 8: original, encrypted and decrypted PII variable, example of one-byte to multiple-byte encryption

The translation method can also be applied in a database. This method involves encrypting the data in Teradata (on a server) before importing it into SAS. This method is ideal when it is preferred that the PHI does not leave the server, and the analyst and programmer do not need the PHI for the use of the data in SAS. The downside is difficulty with decryption and the program must be stored in a secure fashion on the server.

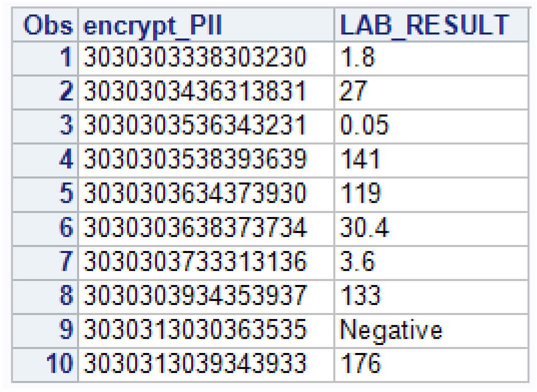
In the following example, the original PII variable is converted to an encrypted variable using the function "char2hexint". Note that there are many of these functions to choose from, with varying levels of complexity. This new encrypted variable is then added to the original table (Output 9). This full dataset remains on the server, and a subset of the data that doesn't include the original PII variable can be extracted from the server, retaining only the non-PII identifying variable (output 10).

```
proc sql;
connect to teradata (tdpid=CONNECTION user=USER_NAME password=PASSWORD);

create table in_db.encrypt_PII as
        select encrypt_PII
                , LAB_RESULT
                from connection to teradata (
                                select PII
                                        , char2hexint(PII) as encrypt_PII
                                        , LAB_RESULT
                                from schema.Table_on_Server_A);

disconnect from teradata;
quit;
```



Obs	PII	LAB_RESULT
1	00038020	1.8
2	00046181	27
3	00056421	0.05
4	00058969	141
5	00064790	119
6	00068774	30.4
7	00073116	3.6
8	00094597	133
9	00100655	Negative
10	00109493	176

Output 9: dataset with original PII variable, kept on the server



Obs	encrypt_PII	LAB_RESULT
1	3030303338303230	1.8
2	3030303436313831	27
3	3030303536343231	0.05
4	3030303538393639	141
5	3030303634373930	119
6	3030303638373734	30.4
7	3030303733313136	3.6
8	3030303934353937	133
9	3030313030363535	Negative
10	3030313039343933	176

Output 10: dataset with encrypted-PII variable that does not contain PII.

CONCLUSION

There are many methods of encrypting using SAS. We have outlined three commonly used methods, and their applications as well as their strengths and weaknesses. It is essential that the SAS programmers select the method that best suits the encryption needs. With the increased proliferation of data collection, and data storage, the need for methods of protecting personal data will continue to increase, and remain essential.

REFERENCES

Department of Labor. 2015. “Guidance on the Protection of Personal Identifiable Information.” Accessed September 22, 2016. https://www.dol.gov/general/ppii

U.S. General Services Administration. 2015. “Rules and Policies – Protecting PII – Privacy Act.” Accessed September 22, 2016. https://www.gsa.gov/portal/content/104256

Luo, S. and Lin, X, 1997. “Using SAS® Bitwise Functions to Scramble Data Fields with Key.” Proceedings of the 22nd Annual SUGI Conference. Available at http://www2.sas.com/proceedings/sugi22/POSTERS/PAPER248.PDF

SAS Institute Inc., 2006. *Base SAS® 9.1.3 Procedures Guide,* Second Edition. Cary, NC: SAS Publishing.

SAS Institute Inc., 2016. “SAS® 9.4 Language Reference: Concepts, Sixth Edition” Available at: http://support.sas.com/documentation/cdl/en/lrcon/69852/HTML/default/viewer.htm#p0ori2cs0xf6ign13vmzmkqgtqap.htm

CONTACT INFORMATION

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

Shuhua Liang
Department of Research and Evaluation
100 S. Los Robles Ave., 5th Floor
Pasadena, CA 91101
(626) 564-3962
shuhua.liang@kp.org

Zoe Bider
Department of Research and Evaluation
100 S. Los Robles Ave., 5th Floor
Pasadena, CA 91101
(626) 564-3904
zoe.bider@kp.org