RMEx 4.0 – Encryption Changes

We have been encrypting sensitive information for close to 10 years. We encrypt all socials, credit card and bank account information. There is a special option to encrypt client account numbers too. The plain text is encrypted and stored in the same file or in a different file. When information needs to be decrypted, we call special programs which convert the encrypted data, back onto plain text. The information can not be accessed without the special programs and the key used to encrypt the data.

We described our encryption as follows.

We offer 128-bit encryption (which is very very strong). Key size can indicate how weak or strong the encryption is. As a general rule, the greater the key size, the better the data is protected (e.g., a 256-bit key generally provides better protection than a 128-bit key). However, this is not always true. For example, data encrypted with the RSA algorithm using a 256-bit key is not as safe as data encrypted using the AES algorithm using a 128-bit key.

We had described our algorithm as the Rijndael Algorithm in Counter mode of operation. Important features of this algorithm were as follows:

- It supported key lengths of 64 bits,128 bits or 256 bits.
- Symmetric Algorithm (Same key was used for encryption and decryption)
- It was free and not patented
- It was the algorithm that was considered the Advanced Encryption Standard (AES)
- It was a block cipher which encrypted 64 bits blocks at a time.
- Counter mode of operation allowed the block cipher to work as a stream cipher. Thus the input length did not have to be a multiple of block size.

Even though we had stated that we were using the 128-bit AES standard, we recently discovered that this was not the case. While data was encrypted, it was not using the standard we has said we were utilizing. As importantly, recent changes have introduced a new requirement of 256-bit encryption for encrypting data in the collection industry. This would also have required major changes to the existing code.

We have taken this opportunity to change our encryption programs and utilize the 256-bit AES standard, which is very strong. This involves changes to many base programs,

and will also affect some custom code. Quantrax will make these changes at no cost to you, as long as the code was written by Quantrax.

If you had previously written custom code (using your own programming team), you will need to change some of the programs that would decrypt any existing encrypted information. The most common use of this was for extracting the consumer social security numbers. The following explains how the changes need to be made with code examples.

Handling custom modifications

We are now using only one C program for the encryption and decryption process.

The encryption and decryption programs are explained below. The programs to call and parameters to be passed are described.

Direct checks

1. Checking account number

Encrypting - Calling the below program with the given parameters will return the encrypted value for the checking account number

ENCRYACC -

XNCLEN 2 (the length.. 18 in this case)

XCKAC# 18 (Checking account number)

ZCKAC# 18 (Checking account number-return value-encrypted)

Company 2 (Comp)

Case number 9 (Case number)

Example:

```
C* ENCRYPT DCKAC#
C                   MOVE      *BLANKS       XCKAC#
C                   MOVE      '18'          ENCLEN         2
C                   If        DCKCOM  <> *zeros
C                   Move      DCKCOM        zzcompn        2
C                   Move      DCKCAS        zzdebtn        9
C                   Else
C                   MOVE      LCOMP         zzcompn
C                   MOVE      LDEBT#        zzdebtn
C                   Endif
C                   CALL      'ENCRYACC'
C                   PARM                    ENCLEN
C                   PARM                    ZCKAC#
C                   PARM                    XCKAC#
C                   PARM                    zzcompn
C                   PARM                    zzdebtn
CSR                 MOVEL     XCKAC#        DCKAC#
```

Decrypting - Calling the below program with the given parameters will return the decrypted value for the checking account number

DECRYACC -

XNCLEN 2 (the length.. 18 in this case)

XCKAC# 18 (Checking account number)

ZCKAC# 18 (Checking account number-return value-decrypted)

Company 2 (Comp)

Case number 9 (Case number)

Example:

```
C* DECRYPT ACCOUNT# (DCKAC#)
C                   MOVE      *BLANKS       XCKAC#         18
C                   MOVE      *BLANKS       ICKAC#         18
C                   MOVE      *BLANKS       ENCLEN          2
C                   MOVE      DCKAC#        ICKAC#
C**** DCKAC#        IFNE      *BLANKS
C                   MOVE      '18'          ENCLEN          2
C                   If        DCKCOM  <> *zeros
C                   Move      DCKCOM        zzcompn         2
C                   Move      DCKCAS        zzdebtn         9
C                   Else
C                   MOVE      LCOMP         zzcompn
C                   MOVE      LDEBT#        zzdebtn
C                   Endif
C                   CALL      'DECRYACC'
C                   PARM                    ENCLEN
C                   PARM                    ICKAC#
C                   PARM                    XCKAC#
C                   PARM                    zzcompn
C                   PARM                    zzdebtn
```

2. Routing number

Encrypting - Calling the below program with the given parameters will return the encrypted value for the routing number

ENCRYROU -

XNCLEN 2

XCKAC# 9

ZCKAC# 9

Company 2

Case number 9

Example:

```
C* ENCRYPT DCKROU
C                   MOVE      '09'          ENCLEN          2
C                   MOVE      *BLANKS       XCKROU
C                   If        DCKCOM  <> *zeros
C                   Move      DCKCOM        zzcompn         2
C                   Move      DCKCAS        zzdebtn         9
C                   Else
C                   MOVE      LCOMP         zzcompn
C                   MOVE      LDEBT#        zzdebtn
C                   Endif
C                   CALL      'ENCRYROU'
C                   PARM                    ENCLEN
C                   PARM                    ZCKROU
C                   PARM                    XCKROU
C                   PARM                    zzcompn
C                   PARM                    zzdebtn
CSR                 MOVEL     XCKROU        DCKROU
```

Decrypting - Calling the below program with the given parameters will return the decrypted value for the routing number

DECRYROU -

XNCLEN 2

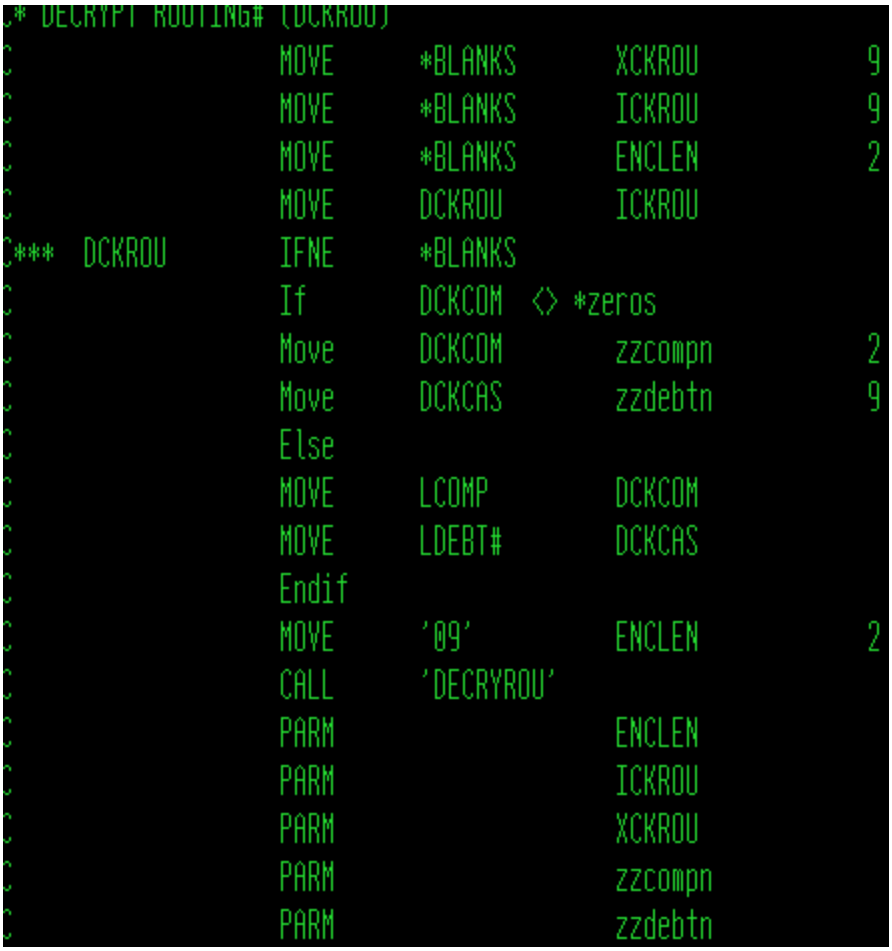
XCKAC# 9

ZCKAC# 9

Company 2

Case number 9

Example:



```
C* DECRYPT ROUTING# (DCKROU)
C                   MOVE      *BLANKS       XCKROU          9
C                   MOVE      *BLANKS       ICKROU          9
C                   MOVE      *BLANKS       ENCLEN          2
C                   MOVE      DCKROU        ICKROU
C***  DCKROU        IFNE      *BLANKS
C                   If        DCKCOM  <> *zeros
C                   Move      DCKCOM        zzcompn         2
C                   Move      DCKCAS        zzdebtn         9
C                   Else
C                   MOVE      LCOMP         DCKCOM
C                   MOVE      LDEBT#        DCKCAS
C                   Endif
C                   MOVE      '09'          ENCLEN          2
C                   CALL      'DECRYROU'
C                   PARM                    ENCLEN
C                   PARM                    ICKROU
C                   PARM                    XCKROU
C                   PARM                    zzcompn
C                   PARM                    zzdebtn
```

Credit cards

1. Credit card number

Encrypting - Calling the below program with the given parameters will return the encrypted value for the credit card number

ENCRYCCN -

XNCLEN 2

XCKAC# 20

ZCKAC# 20

COMP 2

DEBT 9

Example:

```
C                   MOVE      '20'          ENCLEN          2
C                   MOVE      '20'          zzCLEN          2
C                   MOVE      '20'          ENCLEN          2
C     *IN71         IFEQ      *OFF
C                   MOVE      CPCOMP        zzCcmp          2
C                   MOVE      CPDEBT        zzccas          9
C                   Else
C                   MOVE      gcomp         zzCcmp          2
C                   MOVE      gdebt#        zzccas          9
C                   Endif
C                   CALL      'ENCRYCCN'
C                   PARM                    ENCLEN
C                   PARM                    XPCRD#
C                   PARM                    CPCRD#
C                   PARM                    zzCcmp
C                   PARM                    zzccas
```

Decrypting - Calling the below program with the given parameters will return the decrypted value for the credit card number

DECRYCCN -

XNCLEN 2

XCKAC# 20

ZCKAC# 20

COMP 2

DEBT 9

Example:

```
C                   MOVE      '20'          ENCLEN          2
C                   MOVE      CPCOMP        zzCcmp          2
C                   MOVE      CPDEBT        zzccas          9
C                   CALL      'DECRYCCN'
C                   PARM                    ENCLEN
C                   PARM                    CPCRD#
C                   PARM                    QPCRD#
C                   PARM                    zzCcmp
C                   PARM                    zzccas
```

Other sensitive information/ other programs

1. SSN or routing number

Encrypting - Calling the below program with the given parameters will return the encrypted values for the SSN or the routing number

ENCRYALL -

XNCLEN 2

XCKAC# 9

ZCKAC#N 40

Example:

```
MOVE      GSS#          SSN#             9
MOVE      *BLANKS       ENXSS#          40
MOVE      *BLANKS       XSSN#           40
MOVE      '09'          ENCLEN           2
CALL      'ENCRYALL'
PARM                    ENCLEN
PARM                    SSN#
PARM                    XSSN#
MOVEL     XSSN#         ENXSS#
```

Decrypting guarantor SSN - Calling the below program with the given parameters will return the decrypted value for the Guarantor SSN

RTVASSNRN -

COMP 2

DEBT 9

SSN# 9

Example:

```
C*Retrieve SSN FROM FILE SCENCRYP (ENCRYPTED)
CSR   RTVASN       BEGSR
CSR                MOVE      *ZEROS       XXSS#          9 0
CSR                MOVE      *BLANKS      RXSS#          9
CSR                MOVE      GCOMP        PARM1X         2
CSR                MOVE      GDEBT#       PARM2X         9
CSR                MOVE      SSNTYP       PARM3X         3
CSR                CALL      'RTVASSNRN'
CSR                PARM                   PARM1X
CSR                PARM                   PARM2X
CSR                PARM                   PARM3X
CSR                PARM                   RXSS#
CSR                TESTN                  RXSS#            20
CSR   *IN20        IFEQ      '1'
CSR                MOVE      RXSS#        XXSS#
CSR                ENDIF
CSR                ENDSR
```

2. Add and update Guarantor SSN#(Encrypting)

UPDSSNRN -

COMP 2

DEBT 9

SSN# 9

Example:

```
C*SAVE SS# IN FILE SCSSNUM (ENCRYPTED)
C                   MOVE      GSS#          RXSS#           9
C                   MOVE      GCOMP         PARM1X          2
C                   MOVE      GDEBT#        PARM2X          9
C                   CALL      'UPDSSNRN'
C                   PARM                    PARM1X
C                   PARM                    PARM2X
C                   PARM                    RXSS#
C     SPRFU7        IFNE      'Y'
C                   MOVE      *ZEROS        GSS#
C                   END
```

3. Add and update SSN# other than Guarantor

-Encrypting-

UPDASSNN -

COMP 2

DEBT 9

SSNTYP 3 (SSN type*****)

SSN# 9

Example:

```
C*ENCRYPT SPOUSE SSN
C                   MOVE      GSPSS#        XSS#            9
C                   MOVE      GCOMP         PARM1QQ         2
C                   MOVE      GDEBT#        PARM2QQ         9
C                   MOVE      'SPS'         PARM3QQ         3
C                   CALL      'UPDASSNN'
C                   PARM                    PARM1QQ
C                   PARM                    PARM2QQ
C                   PARM                    PARM3QQ
C                   PARM                    XSS#
C     SPRFU7        IFNE      'Y'
C                   MOVE      *ZEROS        GSPSS#
C                   ENDIF
```

-Decrypting-

RTVASSNRN - Retrieve SSN other than Guarantor

COMP 2

DEBT 9

SSNTYP 3 (SSN type*****)

SSN# 9

Example:

```
C*Retrieve SSN FROM FILE SCENCRYP (ENCRYPTED)
CSR   RTVASN       BEGSR
CSR                MOVE      *ZEROS       XXSS#          9 0
CSR                MOVE      *BLANKS      RXSS#          9
CSR                MOVE      GCOMP        PARM1X         2
CSR                MOVE      GDEBT#       PARM2X         9
CSR                MOVE      SSNTYP       PARM3X         3
CSR                CALL      'RTVASSNRN'
CSR                PARM                   PARM1X
CSR                PARM                   PARM2X
CSR                PARM                   PARM3X
CSR                PARM                   RXSS#
CSR                TESTN                  RXSS#            20
CSR   *IN20        IFEQ      '1'
CSR                MOVE      RXSS#        XXSS#
CSR                ENDIF
```

DATA TYPE CODES

We have assigned codes for each type of information stored on one file (SCENCPPT) that keeps track of all of the encrypted information. These codes are as follows, and they have been used in some of the examples above.

001 - SPOUSE SSN

002 - PATIENT SSN

003 - BANKRUPTCY SSN

004 - COMAKER SSN 1

005 - COMAKER SSN 2

006 - COMAKER SSN 3

007 - PURGE FILE SSN

008 - Bankruptcy & Deceased lookup

009 - EMPLOYER FILE SSN

010 - LEGAL SSN 1

011 - LEGAL SSN 2

012 - LEGAL SSN 3