Encryption with DB2 Field Procedures in IBM i 7.1

LINOMA SOFTWARE

Agenda

Presenter:

- Bob Luebbe – Chief Architect for Linoma Software

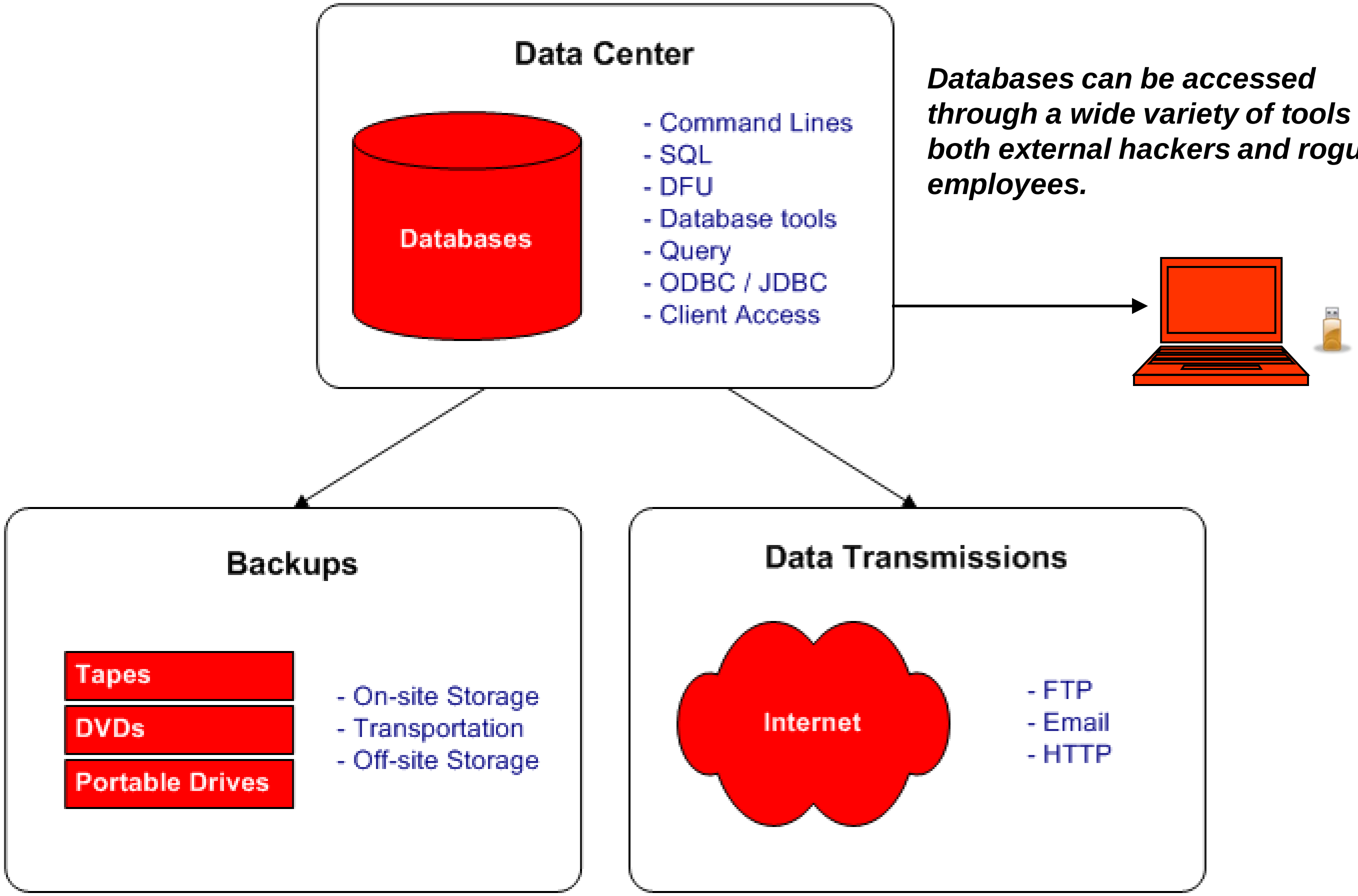
Agenda:

- Data Risks and Trends
- Traditional methods for field (column) encryption
- Introduction to DB2 Field Procedures (FieldProcs)
- How FieldProcs work
- How to get started with FieldProcs
- Potential "gotchas" with FieldProcs
- Performance considerations
- FieldProc program source example
- Feel free to ask any questions

LINOMA
SOFTWARE

Data Risks

Exposed Data



Databases can be accessed through a wide variety of tools by both external hackers and rogue employees.

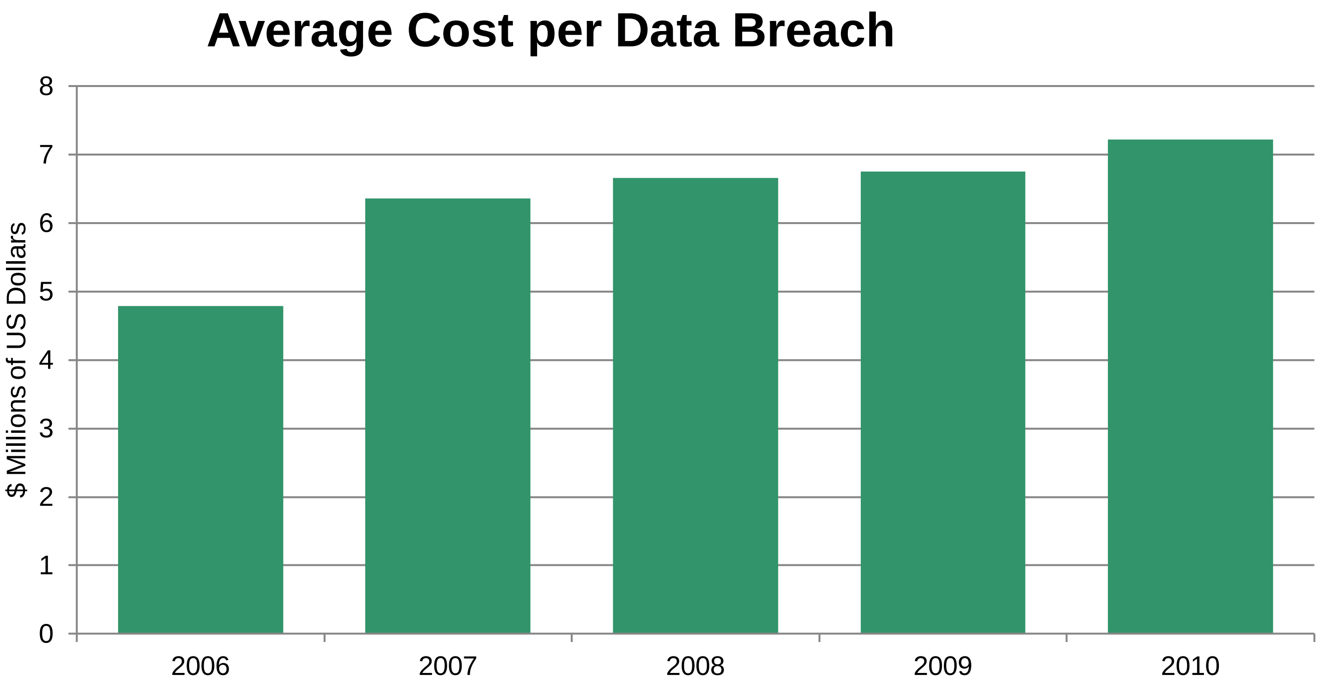
Backup media often passes through many hands to reach its off-site storage location.

Unless otherwise protected, all data transfers travel openly over the Internet and can be monitored or read by others.

LINOMA SOFTWARE

Costs of Losing Data

- *Cost of a Data Breach* study conducted by the Ponemon Institute
- 33,000 lost customer records per incident
- Average cost is now $214 for each lost record
- $7.2 million average organizational cost per breach
- Costs: Administrative and IT labor, Notifications to customers, Public relations, Regaining trust, Lost business



Data Which Needs a High Level of Protection

- Anything that is confidential to the organization, its employees and its customers
- Credit card numbers MasterCard VISA AMEX DISCOVER
- Personal Identifiable Information (e.g. Social security numbers)
- PIN numbers
- Payroll information (e.g. wages)
- Health-related information (PHI)
- Bank Account numbers
- Driver License numbers
- Financial data
- Trade Secrets (e.g. product formulas)

Why Should You Protect This Data?

- To comply with regulations:
  - HIPAA
  - Sarbanes Oxley
  - Gramm-Leach-Bliley Act
  - State privacy laws
- To avoid potential penalties and lawsuits
- To comply with PCI Security Standards MasterCard VISA AMEX DISCOVER
- To avoid bad public relations
- To ensure your continued employment (you don’t want to be the one that “takes the fall”)

> “The personal information of 13,000 individuals who had filed compensation claims with BP after last year's Gulf of Mexico oil spill may have been compromised...
>
> Numerous encryption technologies are available these days to mitigate the risks involved in the loss of a computer or other device, but many companies still don't use them.”
>
> *(Source: ComputerWorld, March 2011)*

Encryption Basics

- Encryption is the process of transforming plain text into cipher text (understandable text becomes unintelligible)
  - **Before:** 4600636500982212
  - **After:** dËKä°BBYý□\åê·Ñ®

- Encryption hides the meaning of the message, but not its existence

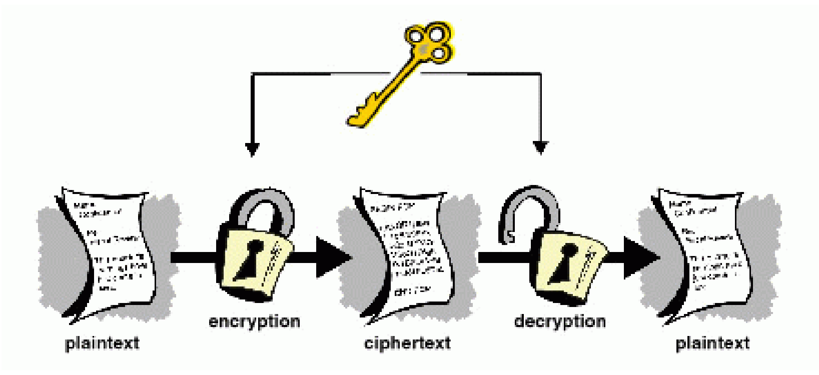
- AES is the most popular encryption Cipher.
  - Approved by NIST
  - No known attacks
  - Fast form of Encryption – 6 times faster than Triple DES
  - Uses symmetric keys
  - Key lengths can be 128, 192 or 256 bits

TERMS

AES is the abbreviation for Advanced Encryption Standard. AES utilizes symmetric key cryptology. It provides strong encryption and is approved by the U.S. Government for protecting top secret information.

Cipher is a pair of algorithms that perform encryption and decryption.

NIST is the abbreviation for National Institute of Standards and Technology. Is a federal technology agency that develops and promotes standards and technology.



LINOMA
SOFTWARE

Encryption Projects

- IBM i shops are at many different stages in their encryption projects:
  - Some have done research, but did not have the time to implement
  - Some used IBM's APIs and built their own custom solution
  - Some used a 3rd party solution
- Compliance requirements (e.g. PCI) generally drive encryption projects
- Encryption projects have been usually limited to extremely sensitive fields like credit card numbers or social security numbers
- Many organizations are re-examining their initial encryption projects (looking for better key management, auditing, security controls, ease-of-use, etc.)
- Organizations would like to encrypt additional sensitive data such as birth dates, names, and other Personally Identifiable Information (PII) data.

Field Encryption Options (before IBM i 7.1)

- For field encryption on IBM i (prior to 7.1), you have two options:
  - Use API calls to encrypt the data before writes and updates (requires program mods)
  - Use column triggers to automatically encrypt data on writes and updates (much better)
- Still need to modify any application where data needs to be decrypted:
  - Screens
  - Reports
  - Batch processes
  - Queries

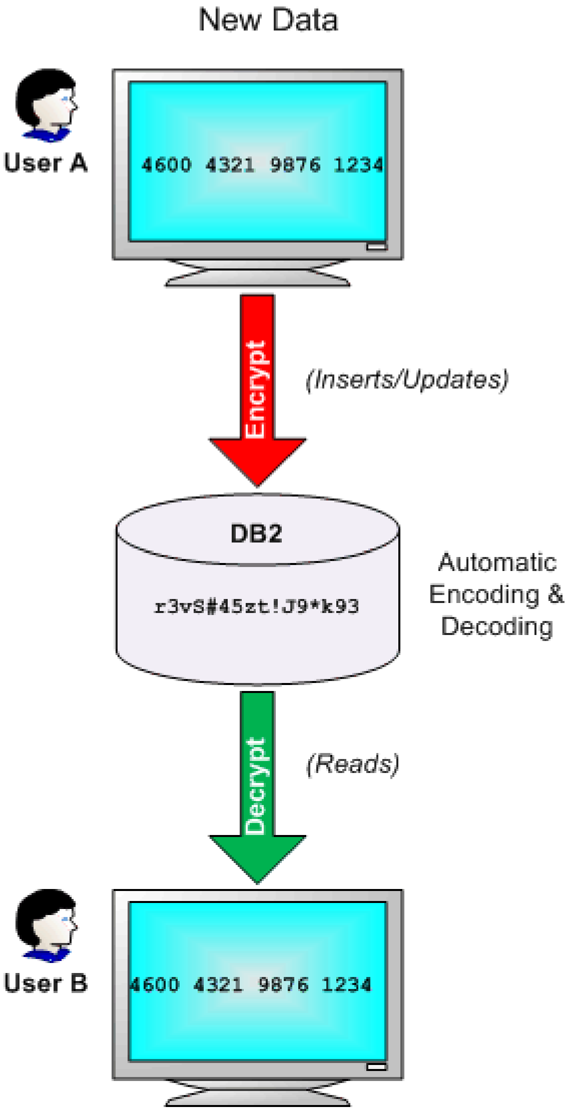
- For numeric fields, have to change database types to alphanumeric OR store the encrypted values in an external (shadow) file
- Could not encrypt date, time and timestamp field types
- What if you don't have the source code or the time?

Field Procedures - Introduction

- Field Procedures (FieldProcs) available in IBM i 7.1
- Linoma has been working with FieldProcs since beta (early 2010)
- FieldProcs are “enabling” technology to simplify encryption projects
- Can minimize or eliminate any application changes
- Stores alternative “encoded” values for fields, so do not need to change your data types, lengths or CCSIDs
- Can be used for a variety of other encoding and decoding schemes:
  - Compression of large strings
  - Normalization of text (41st Street becomes 41st st.)
- Supported for DDS-described physical files and SQL-defined tables
- FieldProcs allowed on multiple fields in a file
- Supported in multi-member files

LINOMA SOFTWARE

Field Procedures - Diagram



LINOMA SOFTWARE

Adding and Removing Field Procedures

Adding a Field Procedure (registering)

- SQL syntax:
  ```
  ALTER TABLE library/filename
      ALTER COLUMN fieldname
      SET FIELDPROC proglib/program
  ```
- No other locks can be on the file while the ALTER statement runs
- Make sure you have *OBJALTER authority to the file, as well as *USE authority to the FieldProc program
- Performs a mass encoding (encryption) of the field values
- May take some time – Submit to batch

Removing a Field Procedure

- SQL syntax:
  ```
  ALTER TABLE library/filename
      ALTER COLUMN fieldname
      DROP FIELDPROC
  ```
- Performs a mass decoding (decryption) of the field values

Encoded Values

- Encoded value can have a different type, length and CCSID than the original field
- Does not change the record format level id – Will not get level checks in programs
- DSPFFD example after adding a FieldProc:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
              Data        Field  Buffer    Buffer        Field    Column
   Field      Type       Length  Length  Position        Usage    Heading
   CMSSNO     CHAR            9       9        43        Both     Social Secur.

     Field text  . . . . . . . . . . . . . . . :  Social Security number
     Coded Character Set Identifier  . . . . . :     37
     Field Procedure Name  . . . . . . . . . . :  CRRP008
       Field Procedure Library . . . . . . . . :  CRYPTO
```

This is the field length that the application will see, but the encoded length may be different.

Shows the name of the program that will be called on the encode and decode operations

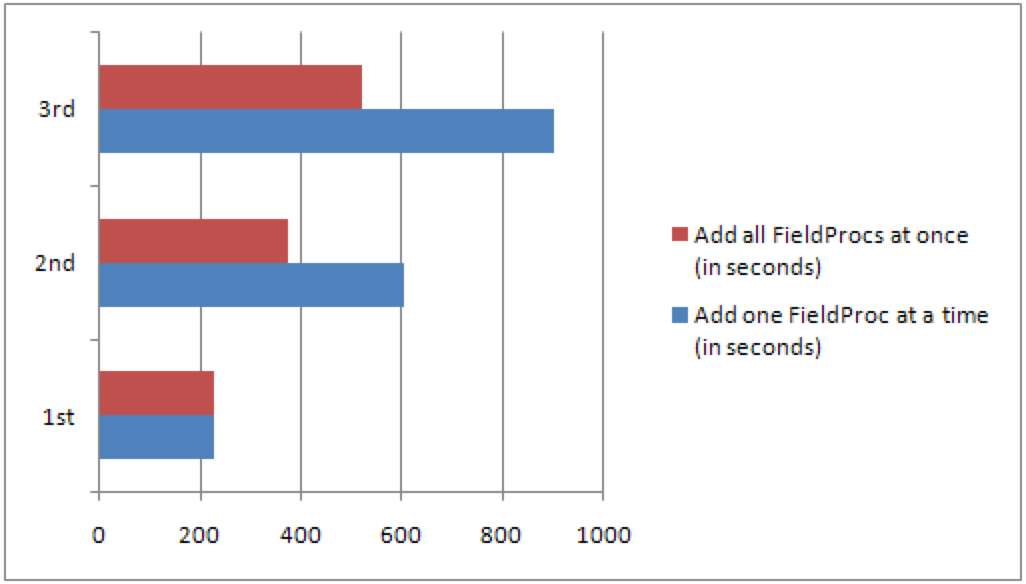
View all FieldProcs on the system with the following SQL statement:

```
  SELECT * FROM QSYS2.SYSFIELDS
```

Each entry will show the file name, field name, type, length and FieldProc name

Performance – Adding Field Procedures

- Time taken (in seconds) to add FieldProcs to a file with 1,000,000 records



- Every time you add a new FieldProc to a file, it runs all the existing FieldProcs on the file to decode and re-encode the values
- Best practice – Use a single ALTER TABLE statement to add all FieldProcs to the file at once

* Tests ran on a model 520. Results are dependent on the size of the IBM i hardware.

Encode Events – Which call the FieldProc

- Native record-level updates and writes
- SQL Insert and Update statements
- Some CL Commands: CPYF, RGZPFM, STRDFU
- Query Processing

```
e.g.  Select SSNO, NAME where SSNO = ‘508773211’
```

On = and <> comparisons (by default) the search is performed using the encoded version of the search value.

* Query search behavior can be changed using the FIELDPROC_ENCODED_COMPARISON setting in the QAQQINI query options file.

- Encoded key values on the SETLL , SETGT, CHAIN, READE:

```
e.g.  SSNO    CHAIN      EMPMAST
```

- Triggers Processing (e.g. for before/after images)

* The FieldProc has no knowledge of what operation (insert, update, etc.) caused the encode or decode event.

Decode Events – Which call the FieldProc

- Native record-levels reads (READ, READE, SETLL, CHAIN, SETGT)
- SQL Select and Fetch

```
e.g.   Select SSNO, NAME where NAME = ‘BOB’
```

- Query Processing

```
e.g.   Select SSNO, NAME, CITY, STATE where SSNO > ‘508773211’
```

On >, >=, <, <= comparisons (by default) all of the existing values in the FieldProc column will be decoded first.

Could be big Performance Hit!

> * Query search behavior can be changed using the FIELDPROC_ENCODED_COMPARISON setting in the QAQQINI query options file.

- Report writers
- File Transfer utilities (e.g. Client Access, Surveyor/400)
- Reading CL commands: DSPPFM, CPYF
- Trigger Processing (e.g. for before/after images)

Query Options file QAQQINI

FIELDPROC_ENCODED_COMPARISON

Specifies the amount of optimization that may be used when queried columns have attached field procedures . *ALLOW_EQUAL is the default behavior.

***NONE** – No optimization to remove field procedure decode option 4 or transformations to optimize field procedure invocations is allowed. For example, the optimizer cannot transform fieldProc(4, column) = 'literal' to column = fieldProc(0, 'literal'). This option is used when the field procedure is not deterministic.

***ALLOW_EQUAL** – Optimization allowed for equal and not equal predicates, GROUP BY, and DISTINCT processing. For example, the optimizer might choose to change the predicate fieldProc(4, column) = 'literal' to column= fieldProc(0, 'literal') in order to facilitate index matching. This option is useful when the field procedure is deterministic but no ordering can be determined based on the result of the field encoding.

***ALLOW_RANGE** – Transformation allowed for MIN, MAX grouping functions, ORDER BY, and all predicates except LIKE in addition to the transformations supported by *ALLOW_EQUAL. This option is useful when the field procedure is deterministic and the encoded value implies ordering

***ALL** – Transformation allowed for all predicates including LIKE, in addition to the transformations supported by *ALLOW_RANGE.

Sorting on Keyed Logicals and Physicals

- It sorts by the encoded (encrypted) value on READs… not by the decoded (decrypted) value

- Example file layout for EMPMAST

```
A           R EMPREC
A             EMPID          7  0       COLHDG(‘Employee id’)
A             NAME          30          COLHDG(‘Employee Name')
A             SSNO           9          COLHDG(‘Social Security Number')
A           K EMPID
```

- Field procedure is added to EMPID

```
ALTER TABLE EMPMAST ALTER COLUMN EMPID SET FIELDPROC PRGLIB/CDRP008
```

- RPG Example of reading entire file

```
 * Read all the employees by employee id
C     *LOVAL        SETLL     EMPMAST
C                   DOW       NOT %EOF(EMPMAST)
C                   READ      EMPMAST
C                   ENDDO
```

Results not ordered:
23233
54332
11111

- Instead, use ORDER BY in an embedded SQL Select statement (if use default QAQQINI option)
  e.g. ***SELECT * FROM empmast ORDER BY empid***

- Should not have problems with CHAINS or READEs

CHGPF Problems

- CHGPF SRCFILE(...) will drop any FieldProcs on a file
- This would perform a mass decryption of the field values!!! WITHOUT WARNING!
- This is a known as an issue by IBM...
- Recommended to submit a request to eliminate this issue with CHGPF (either direct to IBM or through the COMMON Requirement process)
- Alternative:
  - Convert physical file DDS to SQL before using FieldProcs – Then you can use ALTER Table to add, change or remove fields instead of CHGPF
  - Linoma’s Surveyor/400^{tm} or IBM’s i Navigator can help facilitate the conversion to SQL

Stomping on Data

- If conditionally return masked values or blanks, then you may stomp on the data on the subsequent update

- Example file layout for EMPMAST

```
A           R EMPREC
A             EMPID           7  0       COLHDG(‘Employee id’)
A             NAME           30          COLHDG(‘Employee Name')
A             SSNO            9          COLHDG(‘Social Security Number')
A           K EMPID
```

- Field procedure is added to SSNO (to return authorized value; full, masked or none)

```
ALTER TABLE EMPMAST ALTER COLUMN SSNO SET FIELDPROC PRGLIB/CDRP008
```

- RPG Maintenance Program - Example of Stomping over data

```
/FREE
  // Retrieve employee record
  CHAIN EMPID EMPMAST;

  // Prompt for changes to SSNO and NAME
  EXFMT DSP01;

  // Update employee record
  UPDATE EMPMAST;
/END-FREE
```

Decode is performed on SSNO. Masked value of *****1255 is returned.

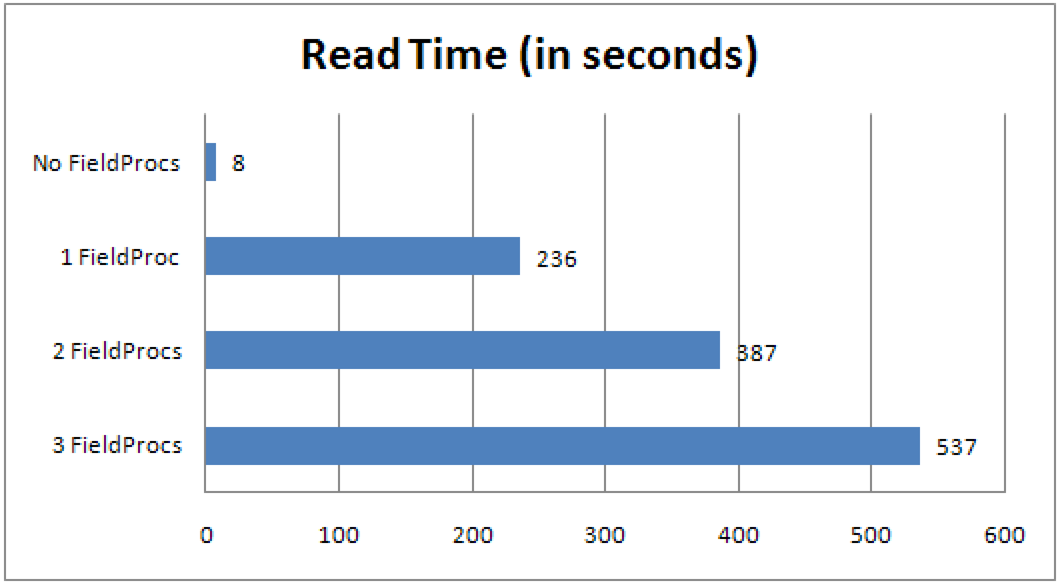
Name and masked Social Security Number is shown on the screen. User just presses enter without making any changes.

Encode is performed on the masked SSNO value. Original value is replaced with *****1255

IBM is working on a PTF that will help enable masking logic in FieldProcs.

Performance – Reading Records

- Test on reading 1,000,000 records:



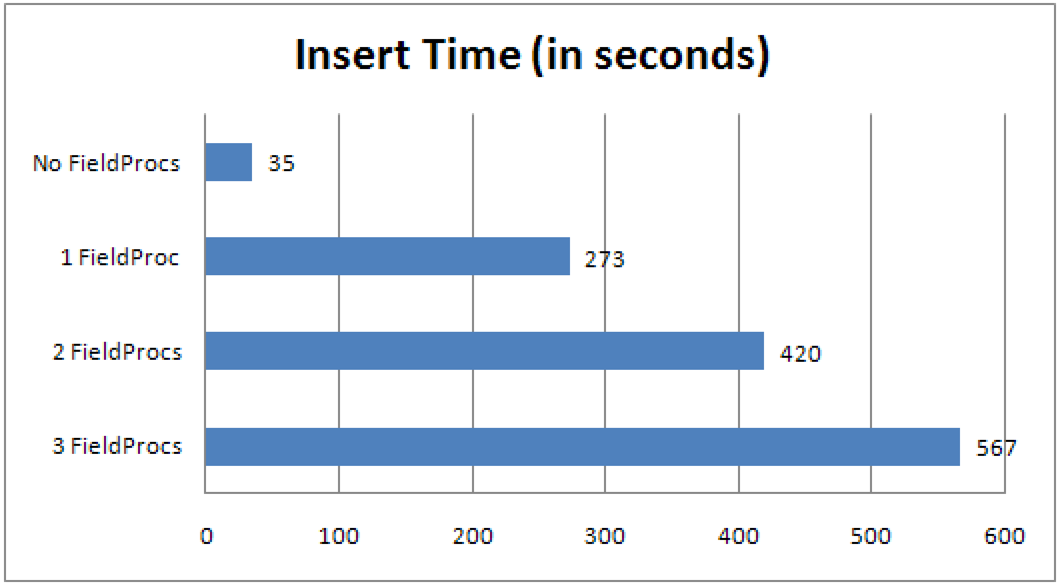
- Test conducted using a READ statement in an ILE RPG program

* Tests ran on a model 520. Results are dependent on the size of the IBM i hardware.

LINOMA SOFTWARE

Performance – Inserting Records

- Test on inserting 1,000,000 records:

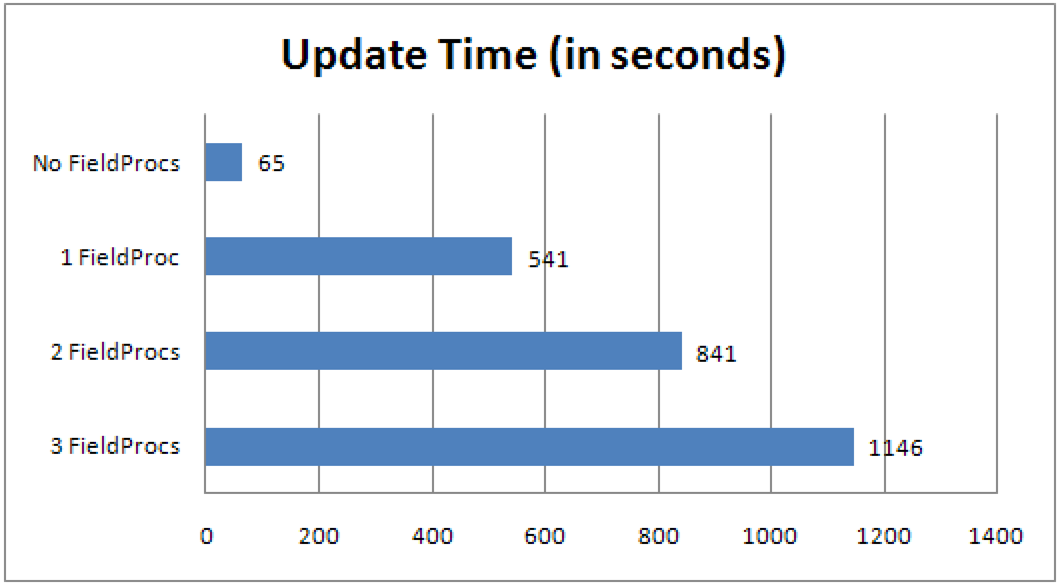


- Test conducted using a WRITE statement in an ILE RPG program

* Tests ran on a model 520. Results are dependent on the size of the IBM i hardware.

Performance –Updating Records

- Tests ran on reading/updating 1,000,000 records



- Test conducted with a READ and UPDATE statement from an ILE RPG program

- On a record UPDATE, the FieldProc encode operation runs even if the field values did not change (unlike a column trigger)

* Tests ran on a model 520. Results are dependent on the size of the IBM i hardware.

Field Procedure Programs

- The FieldProc program must be either written in-house or provided by a vendor (e.g. Linoma)
- FieldProc program criteria:
  - Must be an ILE program
  - Cannot be a Service Program, OPM *PGM or JAVA program
  - Cannot contain any SQL statements
  - Has to be capable of running in a thread (for RPG, specify THREAD(*SERIALIZE) in H spec)
  - Cannot use ACTGRP(*NEW)
- FieldProc program needs 3 sets of logic to process the function code passed in by DB2:

```
// When the FieldProc is added with the ALTER TABLE statement
If Function = 8;
    (logic to return the encoded value's length, type and CCSID to DB2)

// When the data needs to be encoded – primarily called on writes/updates
ElseIf Function = 0;
    (logic to encrypt  the  original value and return it to DB2)

// When the data needs to be decoded – primarily called on reads
ElseIf Function = 4;
    (logic to decrypt  the  encoded value and return it to DB2)
```

FieldProc Program Example (page 1 of 4)

```
H DFTACTGRP(*NO)
H CCSID(*CHAR:*JOBRUN)
H THREAD(*SERIALIZE)

 * Prototype for encrypt function
D encrypt         Pr            16a
D  input                        16a   value
D  length                        5s 0 value

 * Prototype for decrypt function
D decrypt         Pr            16a
D  input                        16a   value
D  length                        5s 0 value

 *****************************************************************
 * Structure describing the parameter data type
D FieldDesc       DS                  qualified
D   sqlType                      5i 0                                      SQL data type
D   byteLen                     10u 0                                      Length of parameter
D   length                      10u 0                                      length in characters
D   precision                    5i 0                                      precision if numeric
D   scale                        5i 0                                      scale if numeric
D   CCSID                        5u 0                                      ccsid
D   allocLen                     5u 0                                      allocated length
D   reserved1                   14a                                        available for use
```

FieldProc Program Example (page 2 of 4)

```
* Function code (0=Encode, 4=Decode, 8=Definition for create time)
D functionCode    S              5i 0
 *  Structure describing optional parameters
D optionalParms   DS                  qualified
D   listLen                     10i 0                                      total length
D   listCnt                     10i 0                                      number of parms
D   parmList                     1a                                        optional parms
* Decoded Data Description
D decodeDataDesc  DS                  LikeDS(FieldDesc)
 * Decoded Data
D decodedData     S              9
 * Encoded Data Description
D encodeDataDesc  DS                  LikeDS(FieldDesc)
 * Encoded Data
D encodedData     S             16
* Returned State for errors
D sqlState        S              5a
 *  Structure for communicating messages.
D messageText     DS                  qualified
D   msgTextLen                   5i 0                                      Msg Length
D   msgTextData               1000a                                        Msg Text

* Entry parms
C     *ENTRY        PLIST
C                   PARM                    functionCode
C                   PARM                    optionalParms
C                   PARM                    decodeDataDesc
C                   PARM                    decodedData
C                   PARM                    encodeDataDesc
C                   PARM                    encodedData
C                   PARM                    sqlState
C                   PARM                    messageText
```

FieldProc Program Example (page 3 of 4)

```
*=============================================================================================*
* MAINLINE                                                                                    *
*=============================================================================================*
/free

 // Handle function requested...
 Select;
 // -----------------------------------------------------------------
 // When ALTER TABLE or CREATE TABLE runs
 // -----------------------------------------------------------------
 When (functionCode = 8);
    // Make sure it is the correct type of Fixed-length character string
    If (decededDataDesc.sqlType <> 452 AND
             decededDataDesc.sqlType <> 453);
       sqlState = '38I02';
       messageText.msgTextData = 'Unexpected data type encountered.‘;
       messageText.msgTextLen = %len(%trim(messageText.msgTextData));
       Return;
    Else;
       // set the encoded Data Attributes
       encodeDataDesc.sqlType   = 452;   // Fixed-length
       encodeDataDesc.byteLen   = 16;    // Size in Bytes of encoded data
       encodeDataDesc.length    = 16;    // Size in Characters of encoded data
       encodeDataDesc.precision = 0;     // precision of numeric parameter
       encodeDataDesc.scale     = 0;     // scale of numeric parameter
       encodeDataDesc.CCSID     = 65535; // CCSID of encoded data
       encodeDataDesc.allocLen  = 0;     // Data length in Variable container
    Endif;
```

FieldProc Program Example (page 4 of 4)

```
 // -----------------------------------------------------------------
 // Encode
 // -----------------------------------------------------------------
 When (functionCode = 0);

    // Encrypt the data
    encodedData = Encrypt(decodedData:9);

 // -----------------------------------------------------------------
 // Decode
 // -----------------------------------------------------------------
 When (functionCode = 4);

    // Decrypt the data
    decodedData = Decrypt(encodedData:16);

 // Error       -- Unsupported option.
 Other;
  sqlState = '38I03';
  messageText.msgTextData = 'Unsupported option requested of ' +     'Field procedure FIELDPROC.';
    messageText.msgTextLen = %len(%trim(messageText.msgTextData));
 EndSl;

 // Normal termination.
 *inLR = *on;
 Return;
/end-free
```

FieldProc Masking (Beta PTF)

- New PTF (in beta) to allow the programmer to return an SQL state '09501' if its determined that the encode is trying to replace a value with a Masked value.

- SQL state 09501 only affects Updates and Inserts through FieldProcs:
  - For an Update operation, 09501 indicates to DB2 that the current value for the column should be used.
  - For an Insert operation, 09501 indicates to DB2 that the default value should be used for the associated column value.

FieldProc (Beta PTF)

- IBM has determined that at certain times they need to the full value returned from the FieldProc. Not masked. One example of this is when they are doing Trigger Processing.

- The beta PTF is introducing a new FieldProc parameter (at the bottom of the *ENTRY PLIST).

- Layout of new parameter (Data Structure):

```
D sqlExtraInfo    DS                 qualified
D   sqlfpInfoLen                  10i 0
D   sqlfpNoMask                    1a
D   reserved                     123a
```

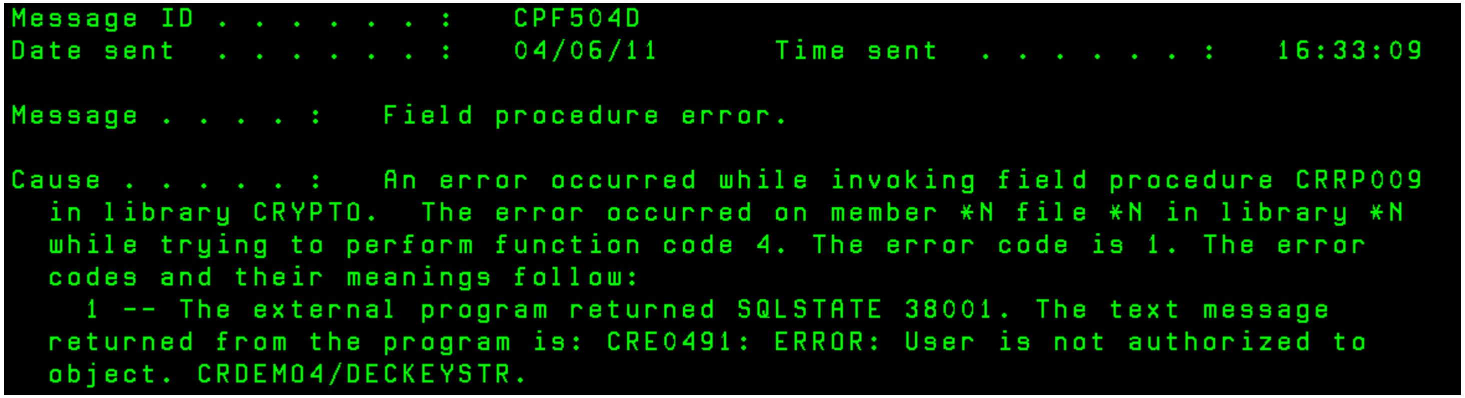
- If a ‘1’ in provided in the sqlpNoMask field, then DB2 wants the full value returned.

- Since this is a new parameter that will be added at the bottom of the *ENTRY PLIST, and some systems may not have the PTF installed yet, you can check for existence of the parameter sent in using the %Parms function

- This should be implemented when using the new masking SQL return code of 09501.

FieldProc Errors

- If you want to report an error back to the calling application, make sure to set the SQLSTATE in your FieldProc before returning
- Valid SQLSTATES are in the 38xxx range
- DB2 will return a message with the id of CPF504D and text of “Field procedure error”.
- You can return additional text in the 2nd level message. Example:



```
Message ID . . . . . . . :   CPF504D
Date sent  . . . . . . . :   04/06/11       Time sent  . . . . . . . :   16:33:09

Message . . . . :   Field procedure error.

Cause . . . . . :   An error occurred while invoking field procedure CRRP009
  in library CRYPTO.  The error occurred on member *N file *N in library *N
  while trying to perform function code 4. The error code is 1. The error
  codes and their meanings follow:
    1 -- The external program returned SQLSTATE 38001. The text message
  returned from the program is: CRE0491: ERROR: User is not authorized to
  object. CRDEMO4/DECKEYSTR.
```

*LOVAL and *HIVAL

- Most RPG shops use *LOVAL SETLL to position to the beginning of a file and *HIVAL SETGT to position to the end of a file

- DB2 will pass *LOVAL to your FieldProc as a string of hex 0s and *HIVAL as a string of hex Fs

- Do not encrypt those special hex values in your FieldProc, or else...

  - DB2 may not position correctly when using *LOVAL or *HIVAL, since it would be trying to position with the encrypted versions of *LOVAL or *HIVAL

  - Subsequent READ/READP operations may not return all records (e.g. if 6 records in the file, you may only get 4)

  - Example of file contents

```
KEY   HEX ENCODED
1     E7D8E95115267BC3EFBABEBF0F318875
2     D0D231AF065105B9FCAC6448143C75C1
3     D36E4FF5DADD020CB8836EDCE578C609
4     F1CC050A407B260D73FB0C9410DB16E8
5     A62292B1273E77A3E822C1BEFF13A8DC
6     D05D1712C41FDC47D53DF38A70CE2961
```

  - Example results of SETGT *HIVAL and READP (if not handled properly):

```
KEY
 2
 6
 5
```

Other Things to Know

- CRTDUPOBJ will duplicate any FieldProcs on the file

- CPYF will always decode the values on the “From” file. It will also encode the values on the “To” file (if FieldProcs exist on the file)

- FieldProc runs in secondary thread. First time a FieldProc is called, it will use the job’s library list. It will not recognize any additional changes to the library list for the job.

- Users must have authority to the FieldProc program
  - They should have at least *USE authority to the FieldProc program OR
  - Create the FieldProc program with USRPRF(*OWNER) and *EXCLUDE public authority. However, this approach will circumvent any authority checks for masking.

- A FieldProc cannot adopt authority from your programs since it runs in a different thread

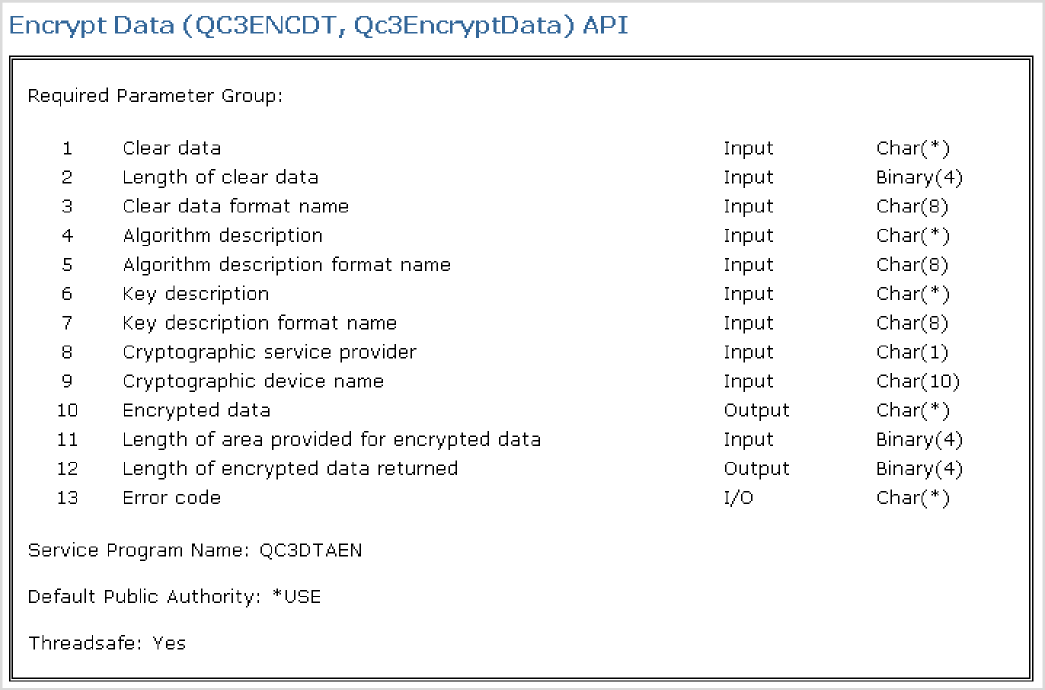
- If a user is not authorized to a FieldProc program, they will get message id CPF4236 with the text of “Not authorized to open member X”.

- FieldProc programs should be “short running” to avoid timeouts. For instance, it is not recommended to perform other file I/O operations in the FieldProc program.

- Make sure to back up FieldProc programs since they are not automatically backed up with the file.

IBM Encryption APIs

- If you want to do it yourself, start by looking at IBM’s APIs of Qc3EncryptData and Qc3DecryptData



Encrypt Data (QC3ENCDT, Qc3EncryptData) API

Required Parameter Group:

1	Clear data	Input	Char(*)
2	Length of clear data	Input	Binary(4)
3	Clear data format name	Input	Char(8)
4	Algorithm description	Input	Char(*)
5	Algorithm description format name	Input	Char(8)
6	Key description	Input	Char(*)
7	Key description format name	Input	Char(8)
8	Cryptographic service provider	Input	Char(1)
9	Cryptographic device name	Input	Char(10)
10	Encrypted data	Output	Char(*)
11	Length of area provided for encrypted data	Input	Binary(4)
12	Length of encrypted data returned	Output	Binary(4)
13	Error code	I/O	Char(*)

Service Program Name: QC3DTAEN

Default Public Authority: *USE

Threadsafe: Yes

- You can read about these APIs at:

  http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0/index.jsp?topic=/apis/qc3encdt.htm
  http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0/index.jsp?topic=/apis/qc3decdt.htm

- We use IBM’s encryption functions in our FieldProcs

Important Elements of a Good Encryption Solution

- Effective Key Management features to meet auditor and compliance requirements
- Stores the Keys securely in encrypted form
- Good security controls on who can create and manage Keys
- Allows only certain individuals to grant/revoke access to decrypted values
- Can restrict users to masked values
- Is easy to rotate Keys (for re-encrypting data)
- Has good audit trails (logs) and security alerts
- Doesn’t entrust too much control to a single person (dual control, separation of duties)
- Shouldn’t be too complex for implementation and on-going maintenance

> *“DB2 field procedure programs can be created by any developer, but please be aware that it takes a deep knowledge of encryption algorithms and best practices to implement a secure encryption solution.”*
>
> **Kent Milligan, IBM**

Field Procedures vs Column Triggers

Benefits of FieldProcs over Column Triggers:

- FieldProcs are faster than Triggers
- FieldProcs can modify data on Read operations – less (or no) program changes
- FieldProcs support the storage of the encrypted (encoded) values in the existing file (this is great when needing to encrypt non-alpha field types)

Benefits of Column Triggers over FieldProcs:

- Column Triggers have been available since V5R1 *(all the kinks have been worked out)*
- Column Triggers only run when the field values change, which may provide better performance

Performance benefit of using APIs (for decryption) versus FieldProcs:

- You can choose when to decrypt the values within your applications. Maybe there are only certain screens or reports where the decrypted values need to be shown.

Summary

- Most shops are not on IBM i 7.1 yet... but some are planning to upgrade soon
- Make sure to get the latest PTFs from IBM before going live
- Still may have to make application changes to fix the sorting problem for keyed fields
- Test, test, test (FieldProcs is new technology):
  - Test all applications that write/update the field
  - Test all applications and queries that read the file
  - Test performance
- Option if not on IBM i 7.1 : can use the column trigger approach to automate encryption
- Additional questions...

Other Resources

- FieldProc article from Kent Milligan (Senior DB2 consultant for IBM):
  (Provides a good overview of FieldProcs, along with a FieldProc RPG source example)

  http://www.mcpressonline.com/database/db2/enable-transparent-encryption-with-db2-field-procedures.html

- IBM i Database SQL Programming Guide:
  (Details on FieldProcs, including parameter descriptions, recommendations)

  http://www-03.ibm.com/systems/i/software/db2/books.html

- IBM i Database Performance and Query Optimization Guide:
  (Indicates when encode/decode operations are ran by DB2, including information on how DB2 optimizes queries that include FieldProc columns. Also provides info on the QAQQINI option.)

  http://www-03.ibm.com/systems/i/software/db2/books.html

How to contact us

LINOMA
SOFTWARE
REVOLUTIONARY SOLUTIONS FOR YOUR WORLD

Web site:	www.linomasoftware.com
E-mail:	bluebbe@linoma.com
Toll-free:	1-800-949-4696
Direct:	(402) 944-4242
Fax:	(402) 944-4243