Encrypting CA IDMS™ Data

Mainframe and Multi-Platform Application Development
MI280SN
Abstract

> Learn how to encrypt data on a CA IDMS database in order to comply with regulatory and audit requirements. This session reviews how Perot Systems successfully implemented a CA IDMS encryption project. Topics include encryption theory, CA IDMS database procedures and project challenges.
Biography

**Terry Schwartz**
Senior Technical Specialist II
Perot Systems

> 31 years data processing experience, including 21 years working with CA IDMS

> CA IDMS experience covers application programming and design, database administration, systems programming, web services, and web connectivity

> Currently Chairperson of the CA IDMS PLC (IUA/EIUA)

> 9th Year at CA World℠
Agenda

> The Driver – Regulatory Requirements
> Encryption History
> Encryption Techniques
> Encryption Implementation
The Driver

Regulatory Requirements
The Driver – Regulatory Requirements

> Health Insurance Portability and Accountability Act of 1996 ("HIPAA")

> Protects "Individually identifiable health information"

> Individually identifiable health information
  - Includes many common identifiers (e.g., name, address, birth date, Social Security number (SSN))

> Privacy rule
  - Define and limit the circumstances in which an individual’s protected health information may be used or disclosed by covered entities
The Driver – Regulatory Requirements

> Sarbanes-Oxley Act of 2002 (SOX)

> Contains 11 titles that describe specific mandates and requirements for financial reporting

> SOX does not specifically reference encryption

> SOX Section 404: Assessment of internal control

  - Requires management and the external auditor to report on the adequacy of the company's internal control over financial reporting (includes IT department and controls)
Encryption History

> Cryptography - the practice and study of hiding information
  - Used as early as the ancient Greeks
  - Julius Caesar used with a shift of 3 to communicate with his generals during his military campaigns

> Encryption - used heavily in WWII (famous Enigma machine used by Germany)

> 1976 US government publishes Data Encryption Standard (DES) specification (56-bit key)

> 2002 US government publishes Advanced Encryption Standard (AES) with key size of 128, 192, or 256 bits
Encryption Techniques
Encryption Techniques
DES

> Algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations

- Expansion
  - The 32-bit half-block is expanded to 48 bits using the expansion permutation, by duplicating some of the bits

- Key mixing
  - The result is combined with a subkey using an XOR operation
  - Sixteen 48-bit subkeys (one for each round) are derived from the main key
Encryption Techniques
DES

> Algorithm operations (cont.)

- **Substitution**
  - After mixing in the subkey, the block is divided into eight 6-bit pieces before processing by the S-boxes, or substitution boxes
  - Each of the eight S-boxes replaces its six input bits with four output bits according to a non-linear transformation, provided in the form of a lookup table
  - The S-boxes provide the core of the security of DES; without them, the cipher would be linear, and trivially breakable

- **Permutation**
  - Finally, the 32 outputs from the S-boxes are rearranged according to a fixed permutation
Encryption Techniques
AES

> Four Rounds

- **SubBytes** — a non-linear substitution step where each byte is replaced with another according to a lookup table.
- **ShiftRows** — a transposition step where each row of the state is shifted cyclically a certain number of steps.
- **MixColumns** — a mixing operation which operates on the columns of the state, combining the four bytes in each column.
- **AddRoundKey** — each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.
Encryption
Implementation
Encryption Implementation
Database Procedures - Encrypt

Application Program:
STORE RECORD

DATABASE Record

IDMSDBMS

Database Procedure Encrypt

CA IDMS Database
Encryption Implementation
Database Procedures - Decrypt

Application Program
- OBTAIN RECORD

IDMS.DBMS

DATABASE Record

Database Procedure Decrypt

CA IDMS Database
Encryption Implementation

> Field level encryption/decryption

> Calc keys can be encrypted (extra code required)

> Encrypting fields involved in index keys – not pretty

> Multi-step implementation
Encryption Implementation

> Database Procedures

- Specified as part of the schema definition
- NO DML commands are allowed
- CA strongly recommends that all database procedures be written in fully reentrant assembler code
- When running in multi-tasking mode REENTRANT database procedures are REQUIRED
Encryption Implementation

> Record Procedures

> Data passed to procedure

- Procedure control block (20 bytes)
- Application control block (236 bytes)
- Application program information block
- Record control block (56 bytes)
- Record occurrence block (length specified in schema)
Encryption Implementation

> Table Driven

<table>
<thead>
<tr>
<th>FLTABLE</th>
<th>DS</th>
<th>0CL24</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>F'2084'</td>
<td>RECORD ID</td>
</tr>
<tr>
<td>DC</td>
<td>F'0'</td>
<td>FIELD DISPLACEMENT</td>
</tr>
<tr>
<td>DC</td>
<td>F'9'</td>
<td>FIELD LENGTH</td>
</tr>
<tr>
<td>DC</td>
<td>F'2221'</td>
<td>RECORD ID</td>
</tr>
<tr>
<td>DC</td>
<td>F'182'</td>
<td>FIELD DISPLACEMENT</td>
</tr>
<tr>
<td>DC</td>
<td>F'9'</td>
<td>FIELD LENGTH</td>
</tr>
</tbody>
</table>

*** Schema record changes require a change to the table
## Encryption Implementation

### Register Usage

<table>
<thead>
<tr>
<th>Register Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>START</strong></td>
<td>LM  R3,R7,0 (R1)</td>
</tr>
<tr>
<td></td>
<td>LOAD PROCEDURE PARMS.</td>
</tr>
<tr>
<td></td>
<td>R3--&gt;PROCEDURE CONTROL BLOCK</td>
</tr>
<tr>
<td></td>
<td>R4--&gt;APPLICATION CONTROL BLOCK</td>
</tr>
<tr>
<td></td>
<td>R5--&gt;COMM BLOCK NOT USED.</td>
</tr>
<tr>
<td></td>
<td>R6--&gt;RECORD CONTROL BLOCK</td>
</tr>
<tr>
<td></td>
<td>R7--&gt;SCHEMA RECORD.</td>
</tr>
</tbody>
</table>

* * *

**Note:**

- The usage of registers R3, R4, and R6 are for accessing control blocks during the encryption process.
- R7 is reserved for the schema record, which is a record containing metadata about the data being encrypted.
Encryption Implementation
Reason for Area Call – Free Storage

BZ RTN NO...NO WORK TO DO
LH R0,PRVERBN R0 = CURRENT VERB.
CH R0,FCN02 FREE STORAGE IF FINISH
BE FREESTO

......
FREESTO LR R1,R11
BAL R8,FREESTG
ST R1,PRUSER CLEAR ADDR OF WORK
B RTN
Encryption Implementation

Check DML Command

NOTAREA  CLC  ERRMIN,STAT00  EXIT IF BAD IDMS STATUS.
BNE     RTN

CLC     PRVERBC(2),STORCDE  IS VERB A STORE?
BE      TIMEBFOR            YES, GO CHECK TIME

CLC     PRVERBC(2),MODCDE   IS VERB A MODIFY
BNE     RTN                 NO, WRONG VERB TYPE
Encryption Implementation
Check Record ID

<table>
<thead>
<tr>
<th>CHKRECID</th>
<th>LH</th>
<th>R15, RECID</th>
<th>RECORD ID OF PASSED REC</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>R14, FRECID</td>
<td>RECORD ID IN THE TABLE</td>
<td></td>
</tr>
<tr>
<td>CR</td>
<td>R15, R14</td>
<td>RECORD IN THE TABLE ?</td>
<td></td>
</tr>
<tr>
<td>BNE</td>
<td>NXTTABLE</td>
<td>NO GO CHECK NEXT TABLE</td>
<td></td>
</tr>
<tr>
<td>BAL</td>
<td>R14, ENCRYPT</td>
<td>YES! GO ENCRYPT THE FIELD</td>
<td></td>
</tr>
<tr>
<td>NXTTABLE</td>
<td>LA</td>
<td>R2, 12 (R2)</td>
<td>GO TO NEXT TABLE ENTRY</td>
</tr>
<tr>
<td>BCT</td>
<td>R8, CHKRECID</td>
<td>OUT OF TABLE ENTRIES?</td>
<td></td>
</tr>
</tbody>
</table>

November 16-20, 2008   Copyright © 2008 CA. All rights reserved.
Encryption Implementation
Point to Field

<table>
<thead>
<tr>
<th>ENCRYPT</th>
<th>ST</th>
<th>R14,REDHOLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>R9,FLDLLEN</td>
<td>LENGTH OF FIELD TO ENCRYPT</td>
</tr>
<tr>
<td>LR</td>
<td>R5,R7</td>
<td>POINT TO SCHEMA RECORD</td>
</tr>
<tr>
<td>MVI</td>
<td>COMPSW,ONSW</td>
<td>ON FOR ODD BYTES</td>
</tr>
<tr>
<td>L</td>
<td>R1,FLDDISP</td>
<td>DISP OF FIELD IN REC (REL TO 0)</td>
</tr>
<tr>
<td>SR</td>
<td>R0,R0</td>
<td></td>
</tr>
<tr>
<td>CR</td>
<td>R1,R0</td>
<td>LENGTH 0?</td>
</tr>
<tr>
<td>BE</td>
<td>NEXTCHAR</td>
<td>NO WE START WITH 1ST BYTE OF REC</td>
</tr>
<tr>
<td>AR</td>
<td>R5,R1</td>
<td>ADVANCE TO DISPL OF FIELD</td>
</tr>
</tbody>
</table>
Encryption Implementation

> Encryption Options – Simple One Key

- Look up character in Table A
- Use Displacement to get character from Encrypt Table B

```
Field
123456789

&dotr%mcs

Key Table A –
abcdefghijklmnopqrstuvwxyz
01234567890@$^@%^&

Key Table B –
+._(*&^%$#@!($@!#$%
POIUYTREW:LKJHGFDSA><m
```
Decryption Module

- Exact same code as encryption in reverse
- Check for GET verb
  
  CLC PRVERBC(2),GETCDE IS VERB A MODIFY

- Must use same key tables as encryption module
Encryption Implementation

> Insert in Schema

MOD

AREA NAME IS VENDOR-A

ESTIMATED PAGES ARE 0

CALL HSLDATE BEFORE FINISH

CALL IMMSDCRP BEFORE FINISH

CALL IMMSECRP BEFORE FINISH

.
Encryption Implementation

> Insert in Schema

MOD RECORD NAME IS D084-VENDPAY-R
SHARE STRUCTURE OF RECORD D084-VENDPAY-R
VERSION 1
RECORD ID IS 2084
LOCATION MODE IS VIA APVENDOR-VENDPAY SET
CALL IMMSDCRP AFTER GET
CALL IMMSECRP BEFORE STORE
CALL IMMSECRP BEFORE MODIFY
WITHIN AREA VENDOR-A OFFSET 0 PERCENT FOR 100 PERCENT
Encryption Implementation

> Process to Implement

- Liberally backup databases
- Modify schema for STORE and MODIFY only
- Run area sweep on records – OBTAIN NEXT then MODIFY
- Modify Schema for GET
- Modify AREA(s) for FINISH
Encryption Implementation

> Challenges

- Testing – Used batch program and copy of area due to abending online task hazardous to your health
- One field was in an index and had to get customer to accept less functionality
- Run the original encrypt twice and you encrypt encrypted values.. start over....
- Print page is your testing friend
Encryption Implementation

> Business Challenges

- Where to keep source code for encryption tables?
- Keep source modules in a separately secured library?
- Removed SSN’s from inquiry screens but still need on update screens
Encryption Implementation

> Performance

- Equivalent area sweeps run with and without decryption
- 560,579 records read in both runs
- Buffers set at 500 and PREFETCH on in both runs
- Jobs run multiple times to validate results
- Calculation – Milliseconds of CPU divided by # records
Encryption Implementation

> Performance

- Without Decryption - 0.01017 milliseconds per read
- With Decryption - 0.01605 milliseconds per read
- 36.6 percent more CPU time per read
- Total 3.3 additional CPU seconds for the 560,579 records read
Summary

> Reasons for Encryption

> Using Database Procedures for Encryption

> Performance

> Challenges
Please Complete a Session Evaluation Form

> The number for this session is **MI280SN**

> After completing your session evaluation form, place it in the basket at the back of the room

  - Please left-justify the session number