Database Procedures for Security

Laura Rochon
Hera Evolution Inc.

CA IDMS User Group - AID

Darmstadt, Germany
September 28th, 2017

Abstract

- This presentation lays out the steps needed to write a database exit to encrypt and decrypt CA IDMS data at a field level. Sample code will be provided as well as all the accompanying external updates such as Schema.

Biography

- **Laura Rochon**
  Hera Evolution Inc.

  Laura has worked with CA IDMS for 30 years, including close to 7 years with Cullinet and CA. Laura is a frequent presenter at CA World and User Conferences in both North America and Europe. As a technical and application DBA, Laura has supported multiple clients in North America, by teaching classes, performing database and system reviews, installation and maintenance, and just normal DBA work. She presently works for Hera Evolution Inc, a leader in CA IDMS Support.

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 heath 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

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

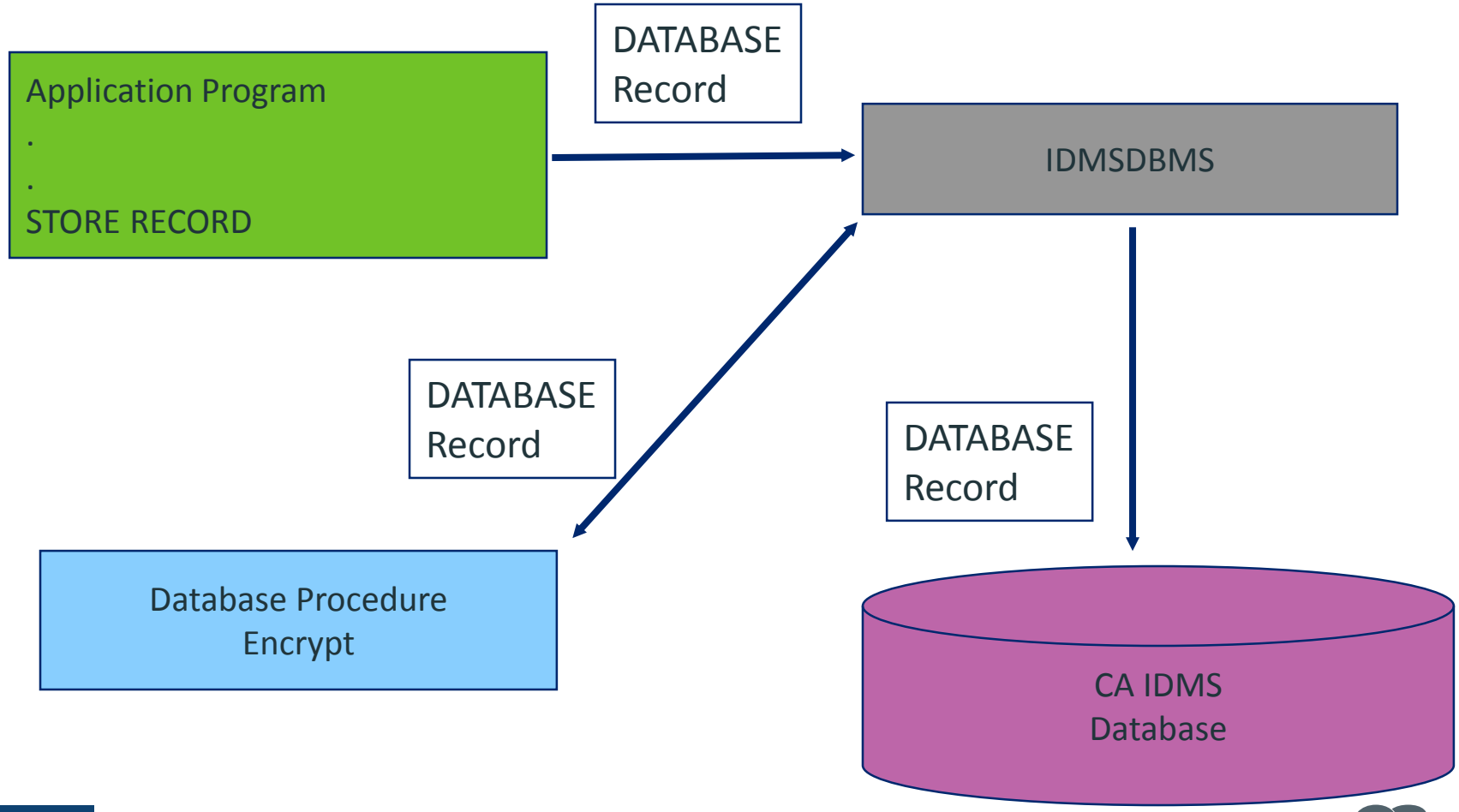
- Algorithm operations (cont.)
  - Substitution
    - After mixing in the subkey, the block is divided into eight 6-bit pieces before processing by the S-boxes, or substitution boxes
    - Each of the eight S-boxes replaces its six input bits with four output bits according to a non-linear transformation, 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

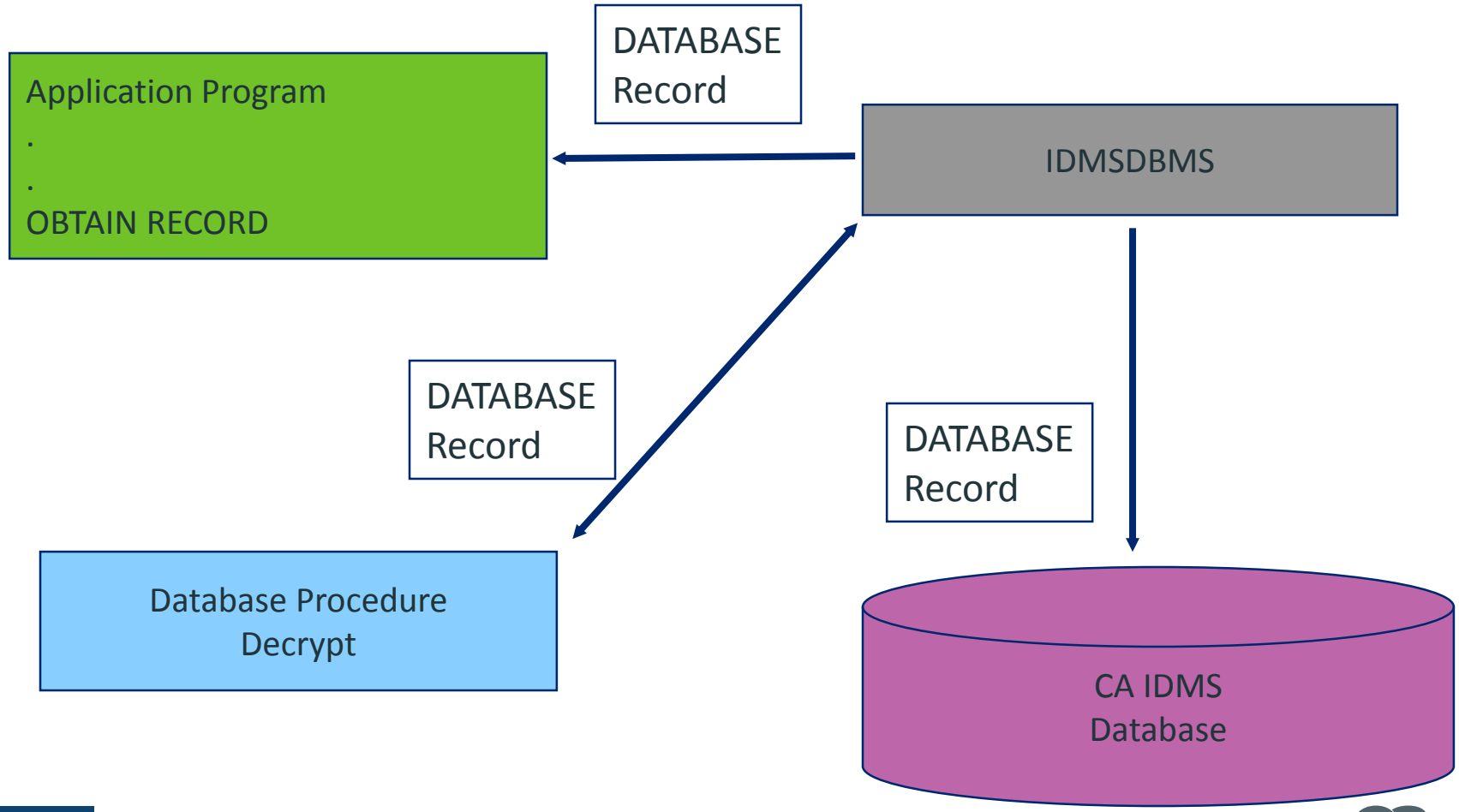
- Four Rounds
  - SubBytes — a non-linear substitution step where each byte is replaced with another according to a lookup table
  - ShiftRows — a transposition step where each row of the state is shifted cyclically a certain number of steps
  - MixColumns — a mixing operation which operates on the columns 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



Encryption Implementation
Database Procedures - Decrypt



Encryption Implementation

- Field level encryption/decryption
- Calc keys can be encrypted
- 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

```
FLTABLE  DS    0CL24
         DC    F'2084'          RECORD ID
         DC    F'0'             FIELD DISPLACEMENT
         DC    F'9'             FIELD LENGTH
         DC    F'2221'          RECORD ID
         DC    F'182'           FIELD DISPLACEMENT
         DC    F'9'             FIELD LENGTH
```

*** Schema record changes require a change to the table

Encryption Implementation
Register Usage

```
START    LM    R3,R7,0(R1)     LOAD PROCEDURE PARMS.

         USING PRCBLK,R3       R3-->PROCEDURE CONTROL BLOCK

         USING APPBLK,R4       R4-->APPLICATION CONTROL BLOCK

*                              R5-->COMM BLOCK NOT USED.

         USING RECBLK,R6       R6-->RECORD CONTROL BLOCK.

*                              R7-->SCHEMA RECORD.
```

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

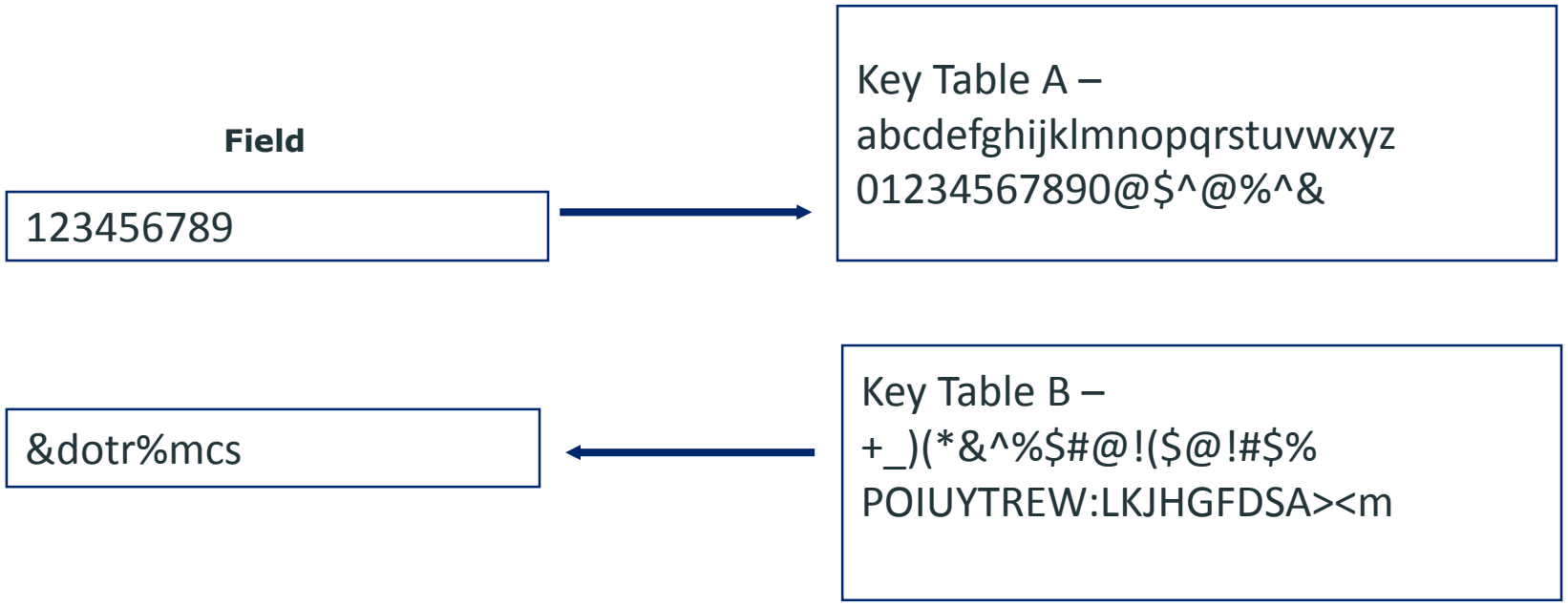
```
CHKRECID LH     R15,RECID          RECORD ID OF PASSED REC
         L      R14,FRECID         RECORD ID IN THE TABLE
         CR     R15,R14            RECORD IN THE TABLE ?
         BNE    NXTTABLE           NO GO CHECK NEXT TABLE
         BAL    R14,ENCRYPT        YES! GO ENCRYPT THE FIELD
NXTTABLE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

Encryption Implementation
Point to Field

```
ENCRYPT     ST     R14,REDHOLD

            L      R9,FLDLEN       LENGTH OF FIELD TO ENCRYPT

            LR     R5,R7           POINT TO SCHEMA RECORD

            MVI    COMPSW,ONSW     ON FOR ODD BYTES

            L      R1,FLDDISP      DISP OF FIELD IN REC (REL TO 0)

            SR     R0,R0

            CR     R1,R0           LENGTH 0?

            BE     NEXTCHAR        NO WE START WITH 1ST BYTE OF REC

            AR     R5,R1           ADVANCE TO DISPL OF FIELD
```

Encryption Implementation

- Encryption Options – Simple One Key
  - Look up character in Table A
  - Use Displacement to get character from Encrypt Table B



Encryption Implementation

- 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

    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 - .01017 milliseconds per read
  - With Decryption - .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