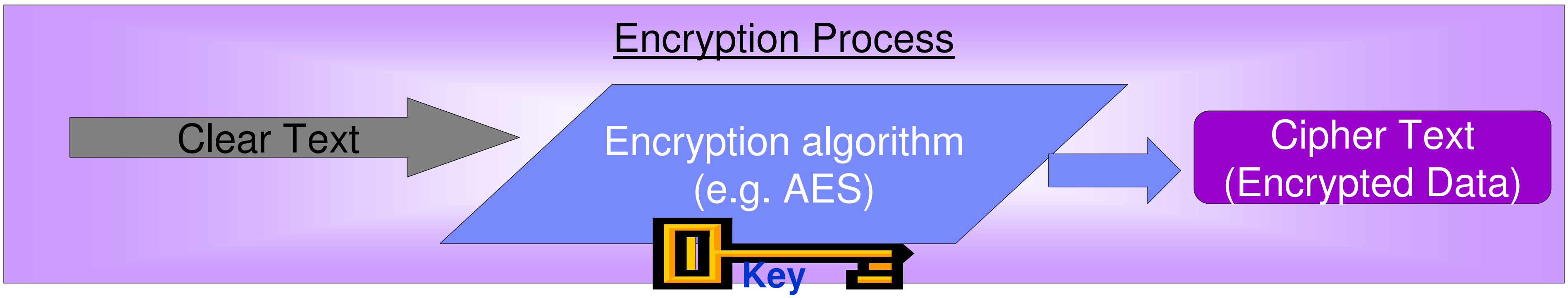
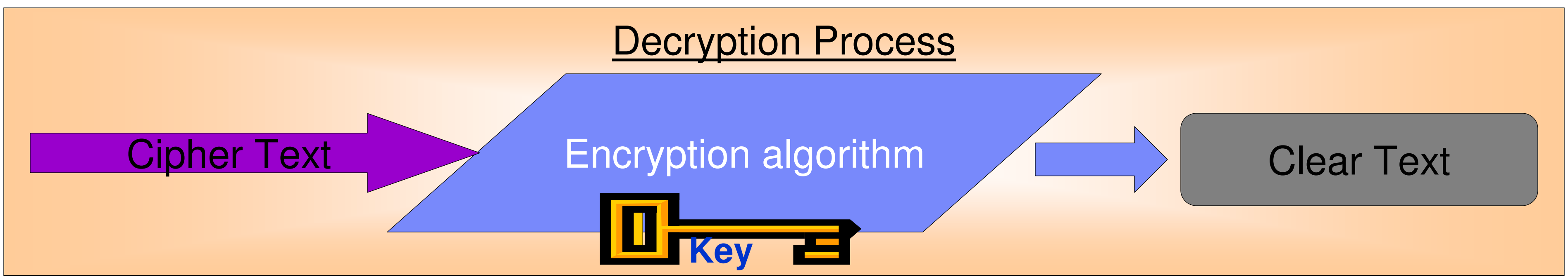
DB2 Encryption on z/OS plus a Real-life Success Story at Verizon Wireless

Ernie Mancill
Executive IT Specialist
IBM zIM Tools

Karl King
Director - Database Administration
Verizon Wireless

Encryption is a technique used to help protect data from unauthorized access





- Data that is not encrypted is referred to as "clear text"
- Clear text is encrypted by processing with a "key" and an encryption algorithm
  - Several standard algorithms exist, include DES, TDES and AES
- Keys are bit streams that vary in length
  - For example AES supports 128, 192 and 256 bit key lengths

Encryption Algorithms – which ones?

- DES
  - Data Encryption Standard – 56 Bit, viewed as weak and generally unacceptable by NIST/FIPS
- TDES
  - Triple Data Encryption Standard – 128 bit, universally accepted algorithm.
- AES
  - Advanced Encryption Standard 128 or 256 bit. Newest commercially used algorithm
- What is acceptable?
  - DES is viewed as unacceptable
  - TDES is viewed as acceptable and NIST compliant
  - AES 128 or 256 is also viewed as acceptable and strategic
- For more information
  - TDES NIST Special Publication 800-67 V1 entitled "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher" and can be found at http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf
  - TDES NIST FIPS Publication 197 entitled "Announcing the Advanced Encryption Standard (AES)" and can be found at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

Hardware Requirements – AES and TDES

– TDES is well supported in both current z9 and z10 hardware combinations

– The tool (5666-P03) will generate exits that can support AES 128, AES 192, or AES 256 keys. However, the type of IBM server determines whether the support for that key length is supported

  – AES 128 support is supported in the hardware (KMC instruction) on z9 and z10.
  – AES 192 and 256 support is supported in the hardware (KMC instruction) on z10 only.
  – AES 256 support is supported in the software (ICSF API) on z9.

– Our suggestion is to implement 128 bit AES on z9 or 256 bit AES on z10 for the best performance experience

Integrated Cryptographic Service Facility (ICSF)

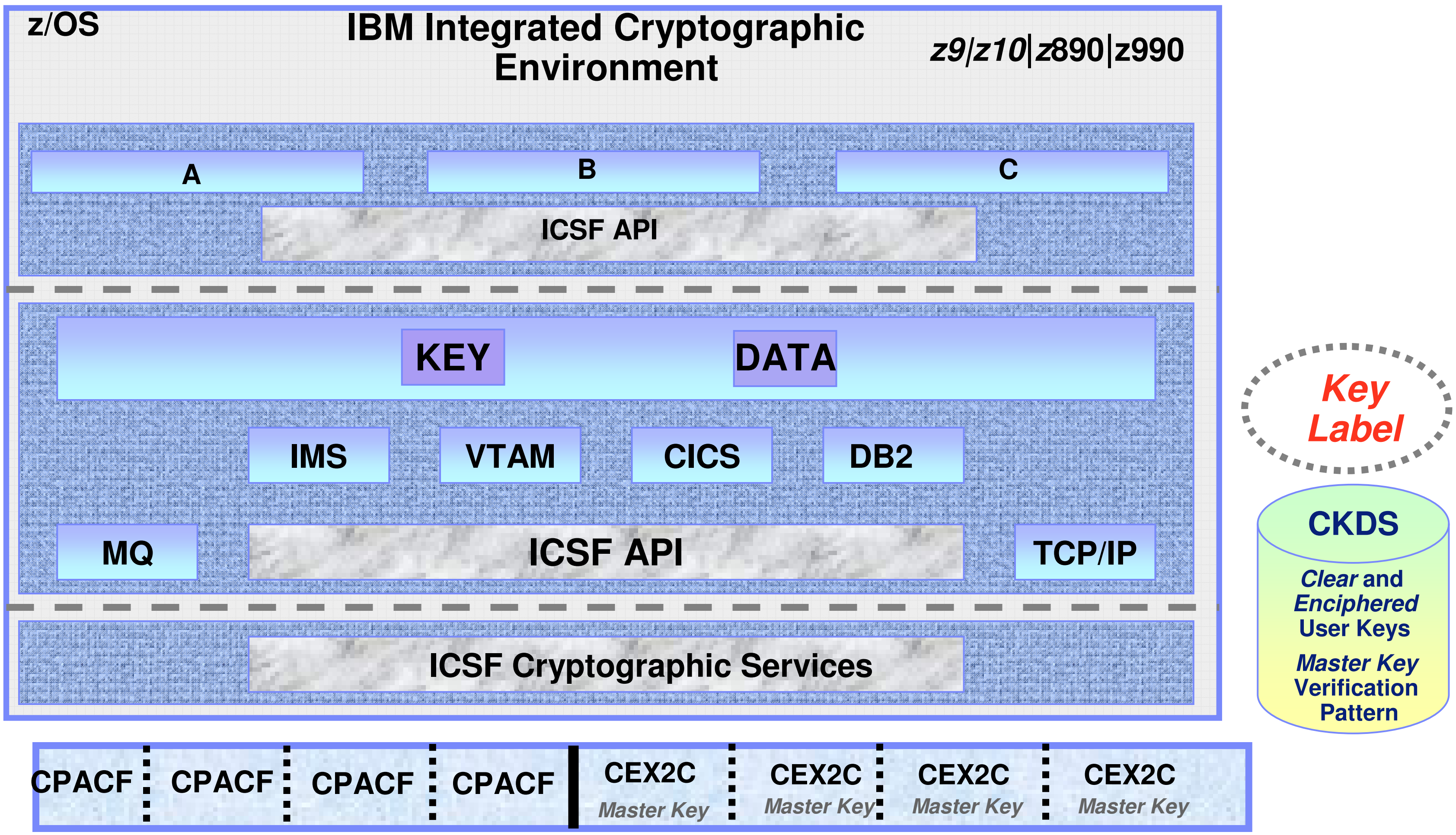
z/OS Integrated Software Support for Data Encryption

- Enhanced Key Management
  (Cryptographic Key Data Set (CKDS) Key Repository)

  Key Creation and Distribution

  Public and Private Keys
  Secure and Clear Keys
  Master Keys

- Access Control for CKDS via Security Access Facility (SAF)
  - Control access to ICSF Callable Services
  - Control access to *Key Labels* (Key Alias) stored in the CKDS
- Hardware and Software Implementation of AES (z9/z10 CPACF)
- Operating System S/W API Interface to Cryptographic Hardware
- Procedures for creating Installation-Defined Callable Services (UDX)



CP Assist for Cryptographic Functions

- Problem State Instructions
- Clear Keys Only
- DES/TDES Encryption
- AES (128 Bit)
- SHA-1 (256 on z9)

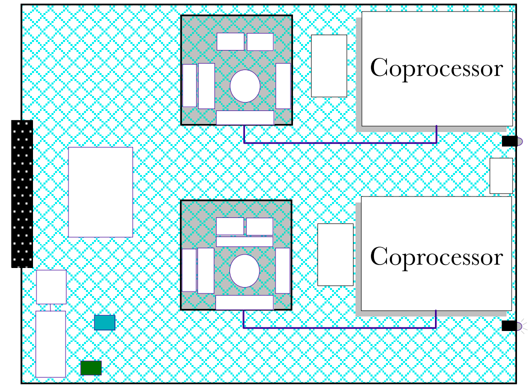
Crypto Express 2 Coprocessor

- ICSF Access Only (Key 0)
- Master Key Stored Within Boundary of Crypto Express 2 Feature
- Secure Key DES/TDES Encryption
- SSL Accelerator
- Tamper Resistant

System z9/z10 Cryptographic Support Summary

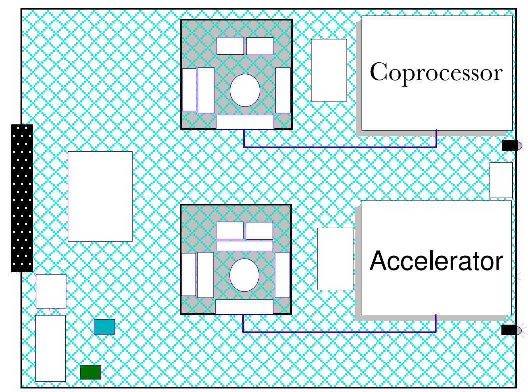
CP Assist for Cryptographic Function (CPACF) “free”

- Supports DES, TDES and SHA-1
- Standard on System z9/z10 (feature code 3863)
- Standard on every CP and IFL
- Advanced Encryption Standard (AES)
- Secure Hash Algorithm – 256 (SHA-256)
- Pseudo Random Number Generation (PRNG)



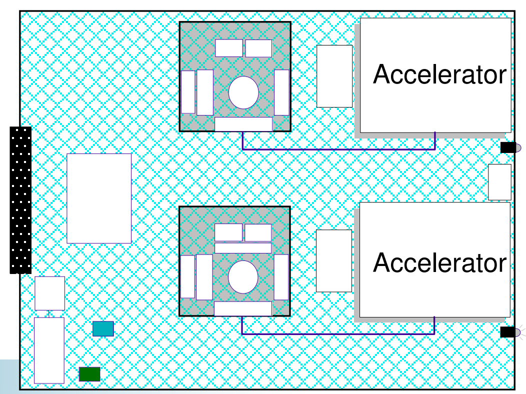
Crypto Express2 (feature code 0853) “fee”

- Two configuration modes
- Coprocessor (default)
- Federal Information Processing Standard (FIPS) 140-2 Level 4 certified
- “Tamper Resistant”
- (Secure Key) – “Exclusive”
- SSL Accelerator (Handshake offload)



Three configuration options

- Default set to Coprocessor (1)
- SSL Acceleration (3)
- Mixture of configuration (2)



What are Keys? (An ICSF Perspective)

- DES Master Key
  - Loaded into the CEX2C hardware, and stored NOWHERE else
  - Used to generate, encrypt, and store user keys into the CKDS (Cryptographic Key Data Set)
- User Keys (Data Encrypting Keys)
  - Generated via ICSF services
  - Used by the IBM Encryption Tool along with encryption algorithm to convert user data to cybertext
  - Stored inside the CKDS
  - Clear or Secure

Secure vs. Clear Key

Secure Key (Highly Secure – High Overhead)

- Keys are encrypted everywhere outside the CEX2C card.
- Interruption will not expose any unprotected key values.
- The data encrypting keys are encrypted in the CKDS with the master key.
- Data encryption/decryption occurs within the CEX2C card.
- AES can now be used for secure key encryption (with HCR7751).

Clear Key (Less Secure – Low Overhead)

- Data encrypting keys are stored in clear text within the CKDS (to improve performance).
- The EDITPROC contains the key label which is passed to the ICSF service which performs a lookup on the CKDS and retrieves the encryption key associated with the key label.
- Once the key has been returned, DB2 retains it in internal memory where it is used for encrypt/decrypt requests by the EDITPROC.
- The CEX2C card is not used to perform actual clear key encrypt/decrypt requests. With HCR7751, a CEX2C card is NOT required for clear key encryption. Prior to HCR7751, a CEX2C was needed because it plays a role in creating a functional CKDS.

Secure Key SQL Performance Results

```
JOBNAME:                     Current Thread Detail             DATE: 06/26/08
DB2 V8 :                                                       TIME: 11:42:17
COMMAND:                                                       CYCLE: MMSS

 CONN ID :                  PLAN     :           CURRENT STATE: INAPP
 CORR ID :                  AUTH ID  :           THREAD START : 11:32:49.4763
 LOCATION:                  SQLID    :           CONN TYPE    : CALL ATTACH
 RQST LOC:                  LUWID    :           [illegible]
 PKG LOC :                  ACCT TKN:
 PKG NAME: FDB2V600.SQLPCRTN.18386E190EC573D6
 +------------ Timings -------------+  +---------- Event Counts ----------+
 ELAPSED:   5:42.97  DB2 ELA:   0:02.96  WAIT      :      27  PACKAGES:     2
 TOT CPU:   0:00.14  DB2 CPU:   0:00.13  IFI       :       0  PARA GRP:     0
 I/O WT :   0:00.00- LOCK WT:   0:00.00  RMT CALL:         0  PARA CPU:     0
 SORT   :   0:00.00- TOT WT :   0:00.30  SORT      :       1  PARA MBR:     0
 NESTED :   0:00.00                      SQL LOGR:         0  DS OPENS:     1
                                         RID LIST:         0
 +---------- SQL Counts -----------+  +------- Buffer Pool/Locking ------+
 TOTAL   :     2054  PREPARES :     1  GETPAGE:       182  MX PG LK:     1
 SELECT  :        1  OPEN CSR :     8  SYNC RD:         9  LOCKESCL:     0
 FETCH   :     2031  INCR BIND:     0  PREFTCH:         4  SUSPENDS:     0
 COMMITS:         2  SECURITY :     0  ASYN RD:         4  TIMEOUTS:     0
 DML     :        0  DDL      :     0  PGS/IO :      14.0  DEADLOCK:     0
```

Clear Key SQL Performance Results

```
DB2 V8 : DTVB                                                       TIME: 11:44:40
COMMAND:                                                            CYCLE: MMSS

 CONN ID :                    PLAN     :           CURRENT STATE: INAPP
 CORR ID :                    AUTH ID :            THREAD START : 11:43:37.9172
 LOCATION:                    SQLID    :           CONN TYPE    : CALL ATTACH
 RQST LOC:                    LUWID    :
 PKG LOC :                    ACCT TKN:
 PKG NAME: FDB2V600.SQLPCRTN.18386E190EC573D6
 +------------- Timings -------------+   +---------- Event Counts ----------+
 ELAPSED:  1:02.64  DB2 ELA:   0:00.36   WAIT      :      13   PACKAGES:    2
 TOT CPU:  0:00.03  DB2 CPU:   0:00.03   IFI       :       0   PARA GRP:    0
 I/O WT :  0:00.01  LOCK WT:   0:00.00   RMT CALL:         0   PARA CPU:    0
 SORT   :  0:00.00- TOT WT :   0:00.33   SORT      :       1   PARA MBR:    0
 NESTED :  0:00.00                       SQL LOGR:         0   DS OPENS:    1
                                         RID LIST:         0
 +----------- SQL Counts -----------+    +------- Buffer Pool/Locking ------+
 TOTAL   :     2054  PREPARES :      1   GETPAGE:        182   MX PG LK:    1
 SELECT  :        1  OPEN CSR :      8   SYNC RD:          3   LOCKESCL:    0
 FETCH   :     2031  INCR BIND:      0   PREFTCH:          4   SUSPENDS:    0
 COMMITS:         2  SECURITY :      0   ASYN RD:          4   TIMEOUTS:    0
 DML     :        0  DDL      :      0   PGS/IO :       26.0   DEADLOCK:    0
```

Some general comments on secure/clear key

- **Clear Key vs. Secure Key Performance**
  - Clear key elapsed time performance is **MUCH** superior than secure key
  - Secure key (performed inside the CEX2C) is generally viewed as more secure from a cryptographic perspective
  - Clear key uses special instructions that run on the z9 – z10 general purpose processors, so performance is measured in milliseconds
  - Secure key encryption is dispatched to run on the cryptographic coprocessors on the CEX2C crypto feature. This tends to be measured in microseconds as this is essentially an I/O operation.
  - Secure key elapsed time measurements (depending on workload and SQL type) can be from 10x to 40x worse than clear key
  - Secure key is probably **NOT** appropriate for most (to date all) OLTP workloads, but each customer needs to make this encryption decision based on their security requirements and performance expectations

Cryptography on z/OS

- Clear Key
  - Key is exposed in the storage of processor
  - Can be viewed in dump of storage
  - If correctly interpreted can expose data
  - Sometimes acceptable for short-lived keys with other constraints
  - Used in software based cryptography
  - Used by CPACF
  - Used by Crypto Express 2 (Configured as CEX2A)

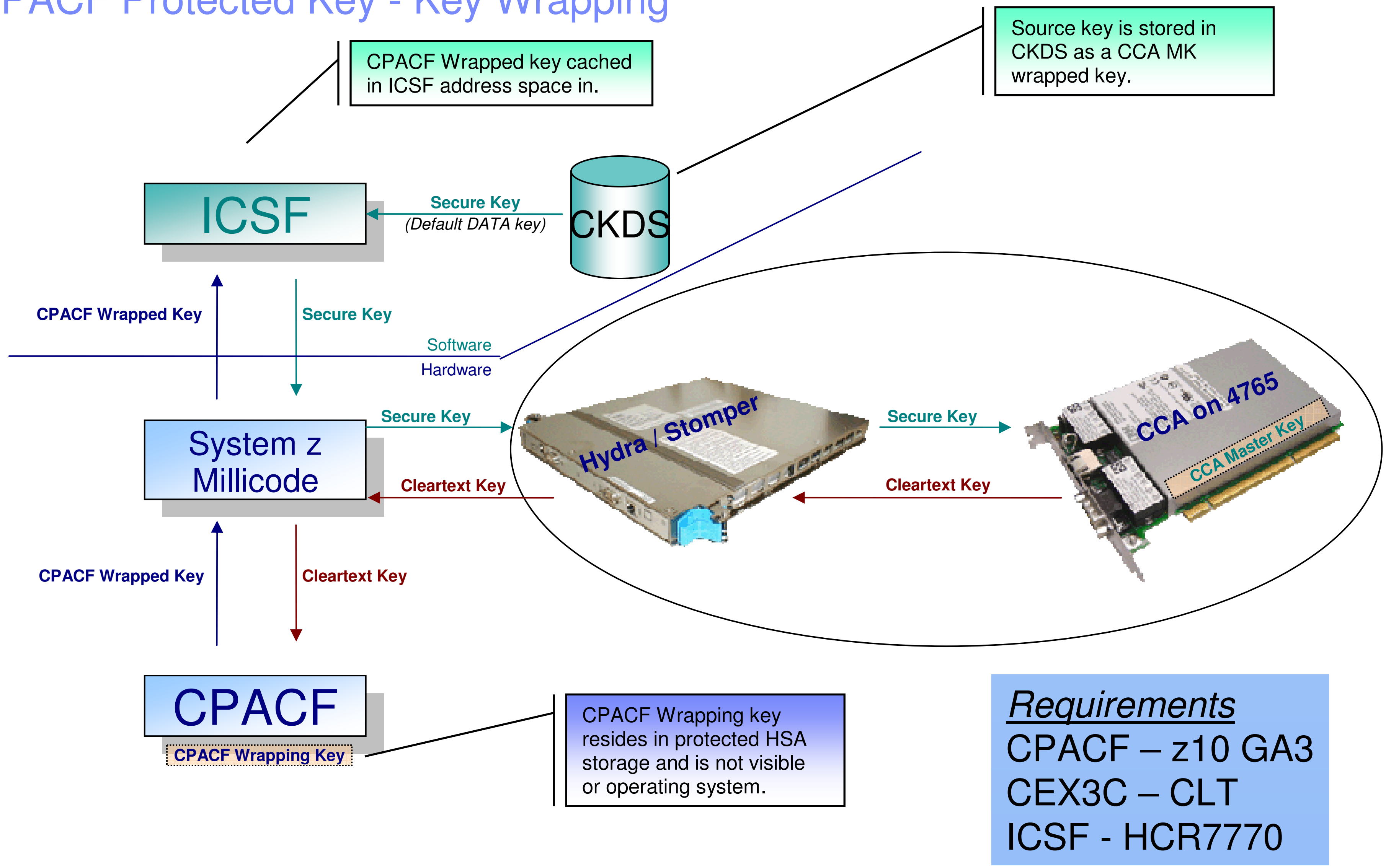
- Secure Key
  - Key is only ever exposed in bounds of a secure processor
  - Can never be seen in storage
  - Dump will not reveal key
  - Key is held encrypted under Master key
  - Crypto Express 2 (Configured as CEX2C) provides this function for System z
  - APIs available via Integrated Cryptographic Support Facility (ICSF)
  - Can be used from Java on z/OS platform

Prior to the introduction of z10/CEX3C protected key option, we recommended Clear key encryption due to performance characteristics of Secure Key, we are now changing that recommendation (if clear key is viewed as “weak”).

System z Symmetric Encryption – Enhanced View

	zSeries 900	System z9 & z10		System z10
	CCA Secure Key	Clear Key	CCA Secure Key	CPACF Protected Key
Key Wrapping: Host Storage	CCA Master Key – **Key material is never visible in the clear outside the tamper resistant hardware boundary**	None – **Key material is visible in the clear in system and application storage .**	CCA Master Key – **Key material is never visible in the clear outside the tamper resistant hardware boundary**	CPACF Wrapping Key – **Key material is not visible in the clear in *operating system or application storage*.**
Key Wrapping: Key Store	CCA Master Key – **Key material is never visible in the clear outside the tamper resistant hardware boundary**	None – **Key material is visible in the clear key store.**	CCA Master Key – **Key material is never visible in the clear outside the tamper resistant hardware boundary**	CCA Master Key – **Key material is never visible in the clear outside the tamper resistant hardware boundary**
Key Store	CKDS or *application key file*	CKDS or *application key file*	CKDS or *application key file*	CKDS only
Encryption Engine	CCF	CPACF **or software**	CEX2C	CPACF
Symmetric Encryption Algorithms	DES and TDES	DES, TDES and AES	DES, TDES and AES	DES, TDES and AES
Benefits	***High Performance*** ***High Security***	***High Performance***	***High Security***	***High Performance*** ***High Security***

CPACF Protected Key - Key Wrapping



DB2 for z/OS and Encryption Exploitation

- IBM Data Server Drivers starting in V9.5 support SSL protocol and AES encryption.
- Starting with Fix Pack 2, non-Java clients supports the Secure Sockets Layer (SSL) protocol. All DB2 Version 9.5 clients now support SSL. In addition, Java and CLI clients now support 256-bit AES encryption.
- SSL connectivity and AES user ID and password encryption requires Communication’s AT-TLS configured and ICSF started. AES support requires PK56287 to be applied on DB2.
- Starting with DB2 for z/OS V8, column level encryption implemented via SQL primitives is supported
- Row level encryption implemented for all supported releases of DB2 for z/OS using the IBM Encryption Tool for IMS and DB2 databases

IBM Data Encryption for IMS and DB2 Databases (5655-P03)

Standard DB2 EDITPROC for Accessing Cryptographic Functions

- All Supported DB2 Versions
- Member of IBM IMS | DB2 Tools Family of Products
- Pre-coded EDITPROC for encryption of DB2® Data
- Encryption/Decryption occurs at the DB2 Row Level
- Unique EDITPROC can be defined for each DB2 Table
- Exploits z/OS Integrated Cryptographic Service Facility (ICSF)
- Exploits zSeries CPACF Cryptographic Hardware Directly
- Requires no changes to your applications
- Fast implementation

Edit Procedures (EDITPROC) are Programs That:

- Transform Data on INSERT | UPDATE | LOAD
- Restore Data to Original Format on SELECT
- Transformations on Entire ROW
- Supported by Utilities
- Implemented via Create Table specification
- Requires unload/load of data

IBM Data Encryption for IMS and DB2 Databases Implementation Summary

Configure the Integrated Cryptographic Service Facility (ICSF)

Enable CP Assist for Cryptographic Functions (CPACF) (z890/z990/z9/z10)
(FC 3863 - This Feature subject to US Export Restrictions)

Install and enable CEX2C (Crypto Express 2) feature
(FC 0863 – Chargeable feature)

Generate and store in the Cryptographic Key Data Set (CKDS) Key Labels

Build the IMS User Exit or DB2 EDITPROC

Generate Data Encryption Key with ICSF ISPF

Obtain Key Label from ICSF Administrator

Use the Sample JCL Provided or the ISPF Panels to generate EDITPROC

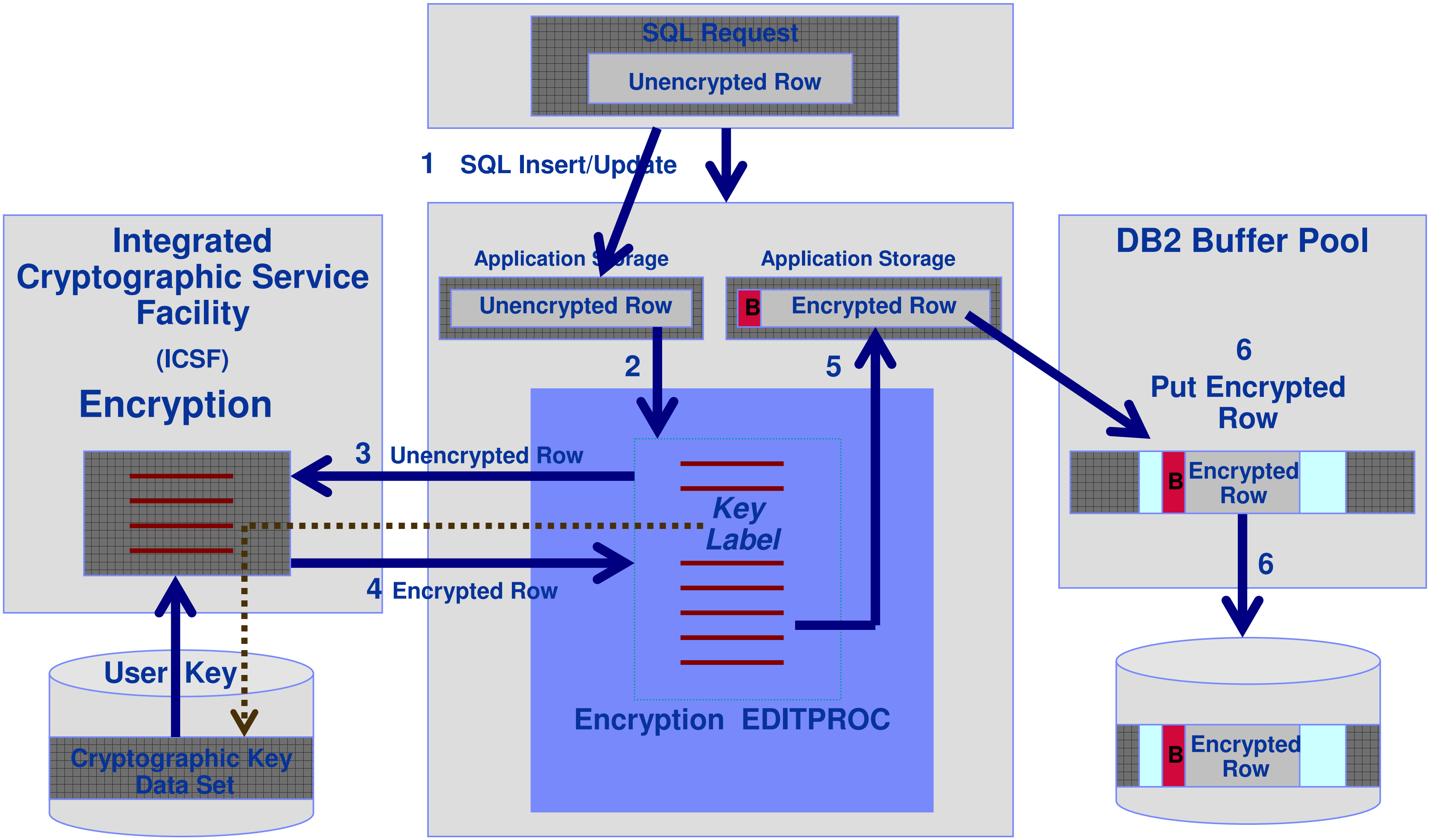
Back - Up and Unload Databases

Create Exits for IMS or EDITPROCS for DB2

Reload the Databases: Data Bases will be Encrypted

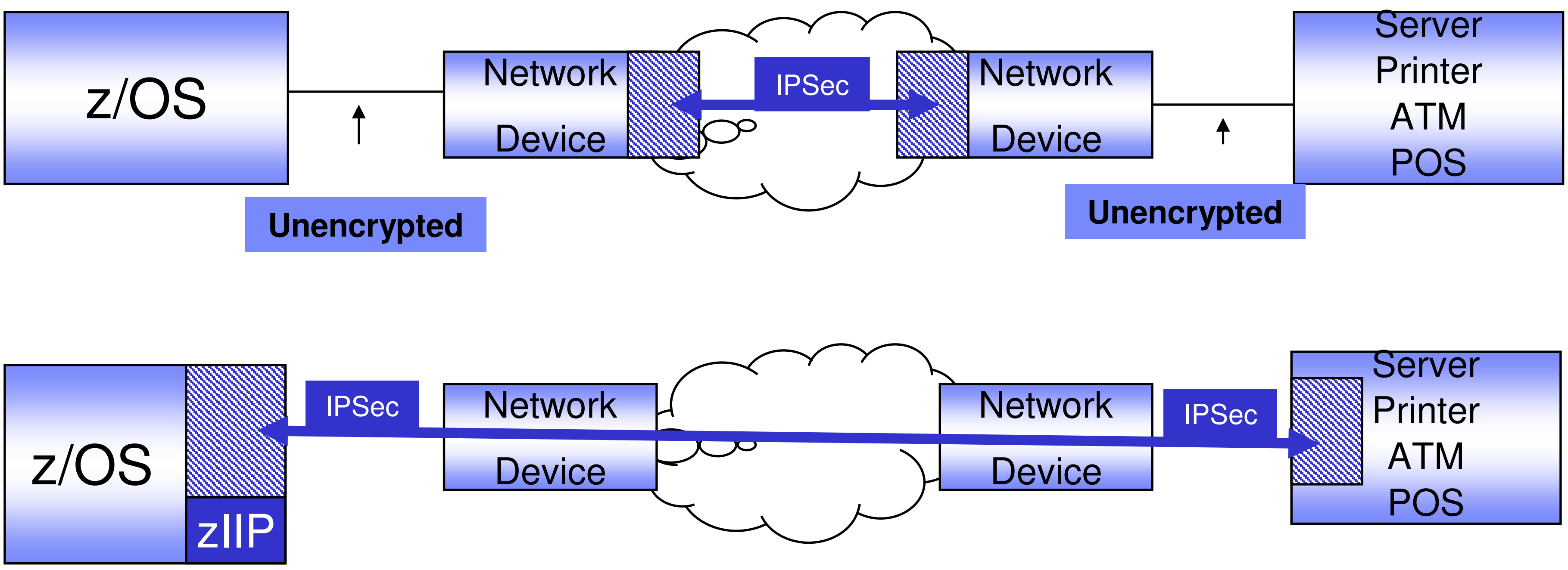
Validate your Output

DB2 Data Encryption Flow – Insert / Update



INFRASTRUCTURE SECURITY

- End-to-end network encryption is becoming more pervasive due to regulatory requirements and data security policies
- Growing requirement for companies that outsource some part of their network and want to control access to confidential data
- zIIP specialty engine support helps reduce the cost of adding IPSec protection



Encrypting your data with Secure Socket Layer support

- DB2 supports Secure Socket Layer (SSL) protocol because it uses the z/OS Communications Server IP Application Transparent Transport Layer service (AT-TLS).
- AT-TLS performs TLS on behalf of the application, such as DB2, by invoking the z/OS system SSL in the TCP layer of the TCP/IP stack.
- To implement SSL support for a DB2 server, the TCP/IP SQL Listener service task of DDF must be capable of listening to a secondary secure port for inbound SSL connections. To specify a secure port to DB2:
  - Specify the TCP/IP port number in the DRDA SECURE PORT field of the Distributed Data Facility Panel 2 (DSNTIP5) during DB2 installation.
  - Update the SECPORT parameter of the DDF statement in the BSDS with the change log inventory (DSNJU003) stand-alone utility
- AT-TLS uses policies that provide system SSL configurations for connections that use AT-TLS. An application continues to send and receive clear text data over the socket while the transmission is protected by the system SSL. AT-TLS support is policy-driven and can be deployed transparently underneath many existing sockets applications.

….. a Real-life Success Story at Verizon Wireless

Agenda

Verizon Wireless' business and database environments.

Evaluation process.

Secure vs. clear key.

Implementation process.

Business Continuity (Disaster Recovery).

Auditing.

Conclusion.

Supplemental Material: ICSF, Key Labels and EDITPROCs.

Verizon Wireless' Business and Database Environments

- Premier wireless carrier.
- 92 million customers and growing.
- Many sales and customer care channels.
- Parallel sysplex with DB2 data sharing.
- DB2 for z/OS V9 NFM.
- 3-tier architecture.
- Billions of DML statements daily.
- Highly tuned systems.
- 99.95 availability requirement. (4.4 hours per year.)
- Complex disaster recovery environment.
- Aggressive data purging and archiving processes.

Agenda

Verizon Wireless' business and database environments.

Evaluation Process

Secure vs. clear key.

Implementation process.

Disaster Recovery.

Auditing.

Conclusion.

Supplemental Material: ICSF, Key Labels and EDITPROCs.

Initial Concerns

- What is the right database encryption solution?
- Would the application need to be modified?
- Would application performance be impacted?
- Which group will own key management?
- What is the security team's role?
- What is the audit team's role?

Database encryption is not just a DBA activity

Evaluation–Scope

- Data model analysis to determine database and backend program impact:
  - Tables with sensitive customer data.
  - Indexes with sensitive customer data and connecting data.
  - Programs, utilities and tools accessing the tables.
- Analysis to determine encryption scope:
  - Backend applications.
  - Middleware applications.
  - Customer facing application.
  - Files on disk and tape.
  - Data transmitted to strategic partners.

Evaluation–Example

➔ Sanitized view of the spreadsheet depicting the tables and programs which access the targeted tables.

Tablespace	Table	LRECL	Npages	Row Count	Compressed	Sensitive column(s)	Edit Proc	Programs	
								Packages	Access Sensitive Columns
XXX	XXX	284	801,023	25,717,923	Y	**Col1**	**XXX**	15	12
XXX	XXX	311	2,227,701	22,276,982	Y	**Col1, Col2, Col3**	**XXX**	387	44
XXX	XXX	195	355,233	10,652,921	Y	**Col2**	**XXX**	26	11
XXX	XXX	63	78,250	5,712,048	Y	**Col6**	**XXX**	2	0
XXX	XXX	134	228,667	5,094,257	Y	**Col1**	**XXX**	17	7
XXX	XXX	141	201,280	4,300,000	Y	**Col3**	**XXX**	16	10
XXX	XXX	131	66,073	4,020,814	Y	**Col1**	**XXX**	30	18
XXX	XXX	82	51,592	3,321,203	Y	**Col2**	**XXX**	11	3
XXX	XXX	97	41,469	2,893,128	Y	**Col2**	**XXX**	**15**	**12**
XXX	XXX	82	20,860	1,600,000	N	**Col4**	**XXX**	6	4
XXX	XXX	146	14,117	1,200,000	N	**Col5**	**XXX**	9	5
XXX	XXX	708	22,252	754,056	N	**Col6**	**XXX**	4	n/a
XXX	XXX	46	4,253	263,681	N	**Col4**	**XXX**	7	7
XXX	XXX	246	143,968	143,968	N	**Col4**	**XXX**	111	8
XXX	XXX	538	6,543	35,330	N	**Col4, Col6**	**XXX**	7	6
XXX	XXX	264	952	11,418	N	**Col5**	**XXX**	10	5
XXX	XXX	45	92	5,458	N	**Col5**	**XXX**	8	4
XXX	XXX	140	55	1,359	N	**Col1**	**XXX**	3	3

Evaluation–Alternatives

We reviewed the various alternatives:

– DB2 V8 column level encryption.

– RSA encryption.

– IBM Encryption Tool for IMS and DB2 Databases:
  - Crypto Express2 cards.
  - IBM Encryption Tool for IMS and DB2 Databases.
  - ICSF.
  - DB2 EDITPROCs.

Implementation would be fast

No program changes

Key management is simplified

The entire table is encrypted

image copies and logs are encrypted

Recovery and DR are manageable

We selected IBM Encryption Tool for IMS and DB2 Databases with EDITPROCs.

Database Administration Concerns

- Should we use secure key or clear key?
- Are there local database recovery implications?
- Are there business continuity implications?
- Can we compress tables with EDITPROCs?
- Are indexes encrypted?
- What are EDITPROC management best practices?
  - How many do we need?
  - Naming standards?
  - How long are they kept after key rotation?
  - What about archived data kept for many years?
  - Do EDITPROCs change our software release process?

Agenda

Verizon Wireless' business and database environments.

Evaluation Process

Secure vs. clear key.

Implementation process.

Disaster Recovery.

Auditing.

Conclusion.

Supplemental Material: ICSF, Key Labels and EDITPROCs.

Secure vs. Clear Key (again!)

- Secure Key
  - Keys are encrypted everywhere outside the CEX2C card.
  - Interruption will not expose any unprotected key values.
  - The data encrypting keys are encrypted in the CKDS with the master key.
  - Encryption/decryption occurs within the CEX2C card.
  - AES cannot be used for secure key encryption.
- Clear Key
  - Data encrypting keys are stored in clear text within the CKDS (to improve performance).
  - The EDITPROC contains the key label which is passed to the ICSF service which performs a lookup on the CKDS and retrieves the encryption key associated with the key label.
  - Once the key has been returned, DB2 retains it in internal memory where it is used for encrypt/decrypt requests by the EDITPROC.
  - The CEX2C card is not used to perform actual clear key encrypt/decrypt requests. However, a CEX2C card is required for clear key encryption because it plays a role in creating a functional CKDS.

Secure Key SQL Performance Results

```
JOBNAME:                      Current Thread Detail            DATE: 06/26/08
DB2 V8 :                                                       TIME: 11:42:17
COMMAND:                                                       CYCLE: MMSS

 CONN ID :                   PLAN     :           CURRENT STATE: INAPP
 CORR ID :                   AUTH ID  :           THREAD START : 11:32:49.4763
 LOCATION:                   SQLID    :           CONN TYPE    : CALL ATTACH
 RQST LOC:                   LUWID    :
 PKG LOC :                   ACCT TKN:
 PKG NAME: FDB2V600.SQLPCRTN.18386E190EC573D6
 +----------- Timings -----------+  +---------- Event Counts ----------+
 ELAPSED:  5:42.97  DB2 ELA:  0:02.96  WAIT     :      27   PACKAGES:     2
 TOT CPU:  0:00.14  DB2 CPU:  0:00.13  IFI      :       0   PARA GRP:     0
 I/O WT :  0:00.00- LOCK WT:  0:00.00  RMT CALL:        0   PARA CPU:     0
 SORT   :  0:00.00- TOT WT :  0:00.30  SORT     :       1   PARA MBR:     0
 NESTED :  0:00.00                     SQL LOGR:        0   DS OPENS:     1
                                       RID LIST:        0
 +---------- SQL Counts -----------+  +------- Buffer Pool/Locking ------+
 TOTAL   :     2054  PREPARES :     1  GETPAGE:        182  MX PG LK:     1
 SELECT  :        1  OPEN CSR :     8  SYNC RD:          9  LOCKESCL:     0
 FETCH   :     2031  INCR BIND:     0  PREFTCH:          4  SUSPENDS:     0
 COMMITS:         2  SECURITY :     0  ASYN RD:          4  TIMEOUTS:     0
 DML     :        0  DDL      :     0  PGS/IO :       14.0  DEADLOCK:     0
```

Clear Key SQL Performance Results

```
DB2 V8 : DTVB                                                  TIME: 11:44:40
COMMAND:                                                       CYCLE: MMSS

 CONN ID :                  PLAN     :          CURRENT STATE: INAPP
 CORR ID :                  AUTH ID :           THREAD START : 11:43:37.9172
 LOCATION:                  SQLID    :          CONN TYPE    : CALL ATTACH
 RQST LOC:                  LUWID    :
 PKG LOC :                  ACCT TKN:
 PKG NAME: FDB2V600.SQLPCRTN.18386E190EC573D6
 +------------- Timings ---------------+   +---------- Event Counts ----------+
 ELAPSED:  1:02.64  DB2 ELA:   0:00.36     WAIT     :       13   PACKAGES:     2
 TOT CPU:  0:00.03  DB2 CPU:   0:00.03     IFI      :        0   PARA GRP:     0
 I/O WT :  0:00.01  LOCK WT:   0:00.00     RMT CALL:         0   PARA CPU:     0
 SORT   :  0:00.00- TOT WT :   0:00.33     SORT     :        1   PARA MBR:     0
 NESTED :  0:00.00                         SQL LOGR:         0   DS OPENS:     1
                                           RID LIST:         0
 +----------- SQL Counts -----------+      +------- Buffer Pool/Locking ------+
 TOTAL   :      2054  PREPARES :       1   GETPAGE:        182   MX PG LK:     1
 SELECT  :         1  OPEN CSR :       8   SYNC RD:          3   LOCKESCL:     0
 FETCH   :      2031  INCR BIND:       0   PREFTCH:          4   SUSPENDS:     0
 COMMITS:          2  SECURITY :       0   ASYN RD:          4   TIMEOUTS:     0
 DML     :         0  DDL      :       0   PGS/IO :       26.0   DEADLOCK:     0
```

Secure vs. Clear Key: Database Load Results

- Database utility loads of 200,00 rows yielded the following results:

(In seconds)	Clear Key	Secure Key
CPU	2	8
Elapsed	18	259

As you can see from the LOAD and SQL examples, secure key is considerably more CPU intensive.

Agenda

- Verizon Wireless' business and database environments.
- Evaluation Process
- Secure vs. clear key.
- **Implementation process.**
- Disaster Recovery.
- Auditing.
- Conclusion.
- Supplemental Material: ICSF, Key Labels and EDITPROCs.

Implementation–Setup

- Required hardware and software:
  - Crypto Express 2 cards.
  - The IBM Encryption Tool for IMS and DB2 Databases.
  - Integrated Cryptographic Services Facility.
- Have the proper ICSF RACF authorities.
- Create the master keys.
- Create the key label and data keys
- Create the DB2 EDITPROC.
- Create table(s)—using EDITPROC name.

IBM Encryption Tool for IMS and DB2

- Install the *IBM Encryption Tool for IMS and DB2 Databases*. Our DB2 Systems programmers installed it.
- Input is the key label created using the ICSF.
- Output is the EDITPROC. Specify the EDITPROC name to be created by the tool.
- Either the DB2 systems programmer or DBA may run the job to create the EDITPROC.

IBM Encryption Tool for DB2 and IMS—JCL example

Jobcard goes here.

```
//LINK     EXEC PGM=IEWL,PARM='LIST,XREF,RENT'
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SDECLMD0  DD DSN=D                0,DISP=SHR
//SCSFMOD0  DD DSN=CSF.SCSFMOD0,DISP=SHR
//SYSUT1    DD UNIT=SY      CE=(1024,(50,50))
//SYSLMOD   DD DSN=DAP1.DPV1.DSNEXIT(Editproc name goes here),DISP=SHR
//SYSLIN    DD *
  ENTRY DECENA00
  INCLUDE SDECLMD0(DECENA00)
  INCLUDE SCSFMOD0(CSNBKRR)
  NAME Editproc name goes here(R)
//*
//BATCHTSO EXEC P                    =25,REGION=0M,COND=EVEN
//SYSLIB   DD DISP=S                 EXIT
//ISPLLIB  DD DISP=S                 FOR ZAP PGM AMASPZAP **
//ISPPLIB  DD DISP=S
//ISPSLIB  DD DISP=S
//ISPMLIB  DD DISP=S
//ISPTLIB  DD DISP=S
//SYSPROC  DD DISP
//SYSEXEC  DD DISP=SHR,DSN=DAP1.CRYPT.SDECCEXE
//ISPTABL  DD DSN=&&TEMP1,UNIT=SYSDA,SPACE=(CYL,(1,1,25)),
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB),DISP=(NEW,DELETE)
//ISPPROF  DD DSN=&&TEMP2,UNIT=SYSDA,SPACE=(CYL,(1,1,25)),
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB),DISP=(NEW,DELETE)
//SYSTSPRT DD SYSOUT=*
```

IBM Encryption Tool for DB2 and IMS—JCL example

```
//ISPLOG   DD SYSOUT=*,DCB=(BLKSIZE=800,LRECL=80,RECFM=FB)
//SYSTSIN  DD *
  PROFILE PREFIX(USERID)
   ISPSTART CMD(%DECENC02 DB2 Editproc name goes here -
   Key label name goes here                              )
//*
//*----+----1----+----2----+----3----+----4----+----5----+----6----
//* YYYYYYYYYY = ENCRYPTION KEY TO BE USED, E.G., ICSFDB2KEY
//*              CAN BE UP TO 64 CHARACTERS MAXIMUM LENGTH
//*
//*        COPY ENCRYPTION EXIT TO OTHER DB2 MEMBERS.
//*
//COPY1     EXEC PGM=IEBCOPY,REGION=4096K
//SYSPRINT  DD SYSOUT=*
//INDD      DD DISP=SHR,DSN=DAP1.DPV1.DSNEXIT
//OUTDD     DD DISP=[illegible]DSNEXIT
//SYSIN   DD  *
  COPY I=((INDD,R)),O=OUTDD
  SELECT MEMBER=(Editproc name goes here)
//*
//*        COPY ENCRYPTION EXIT TO OTHER DB2 MEMBERS.
//*
//COPY2     EXEC PGM=IEBCOPY,REGION=4096K
//SYSPRINT  DD SYSOUT=*
//INDD      DD DISP=SHR,DSN=DAP1.DPV1.DSNEXIT
//OUTDD     DD DISP=SHR,DSN=DAP1.DPV3.DSNEXIT
//SYSIN   DD  *
  COPY I=((INDD,R)),O=OUTDD
  SELECT MEMBER=(Editproc name goes here)
```

Implementation–Verification

- Create/load an encrypted and clear text version of a table.
- DSN1PRNT a few pages of both tables to display the clear text and encrypted contents.
- Execute all tools and utilities against the encrypted table to verify they work as expected.
- Benchmark “heavy lifting” utilities against both tables to track execution information.
- Execute the busiest batch and CICS transactions against both tables.
- Measure using DB2 traces and performance tools.

We collaborated with the IBM team every step of the way. This was a team effort.

Implementation–Example

Encrypted Table		Utillity		Clear Text Table	
Cpu Time	Elapsed Time			Elapsed Time	Cpu Time
00:01:46.02	00:01:37.86	3rd party	Unload	00:01:42.64	00:01:08.31
00:03:55.04	00:06:25.53	IBM	Unload	00:04:39.11	00:03:06.85
00:03:45.13	00:04:07.73	3rd party	Load	00:03:40.55	00:03:12.89
00:06:44.83	00:15:28.67	IBM	Load	00:14:50.57	00:05:40.45
00:03:33.44	00:19:56.46	3rd party	Reorg	00:05:49.17	00:02:12.37
00:12:03.00	00:20:35.25	IBM	Reorg	00:26:25.80	00:09:16.09
00:01:32.04	00:03:50.03	3rd party	IX Rebuild	00:01:20.30	00:00:48.94
00:01:59.83	00:01:48.08	IBM	IX Rebuild	00:02:23.41	00:01:07.70
00:00:08.10	00:07:05.19	3rd party	Image Copy	00:03:51.43	00:00:07.56
00:00:33.58	00:25:13.49	IBM	Image Copy	00:10:01.52	00:00:33.77
00:05:25.35	00:06:27.12	IBM	Runstats	00:05:04.25	00:04:00.82

Results are from our system.
Your mileage may vary.

Implementation–Final

- After thorough testing and measuring, we…
  1. encrypted our development environments.
  2. encrypted our test environments.
  3. encrypted our smallest production system.
  4. implemented in the remaining production systems.
- With each of the above implementations, we measured and tracked database utility, batch and on-line performance.
- We observed that the encryption overhead was consistent with compression.
- It has been over two years since we completed the database encryption project and we have not encountered any issues.

Agenda

Verizon Wireless' business and database environments.

Evaluation Process

Secure vs. clear key.

Implementation process.

Disaster Recovery.

Auditing.

Conclusion.

Supplemental Material: ICSF, Key Labels and EDITPROCs.

Business Continuity (Disaster Recovery)

- Attempts to run SQL against encrypted tables before entering the master key:
  -652 VIOLATION OF INSTALLATION DEFINED EDIT OR VALIDATION PROCEDURE proc-name.
  Explanation: The result of the SQL statement has been rejected by the installation defined edit or validation procedure 'proc-name' for the object table.
- Perform master key and CKDS initialization:
  - Display the coprocessors from coprocessor management.
  - Enter master key in each LPAR—using your production master key.
  - Enter the CKDS/PKDS data set names in quotes.
  - Set Initialize the CKDS and PKDS? (Y/N) to **N**.
  - Set Initialize new online coprocessors only? (Y/N) to **N**.
  - Display the coprocessors from coprocessor management.
  - Refresh the in-storage CKDS.
  - You should get an *Initialization Complete* message in the upper right hand corner.

Post—Disaster Recovery

- Delete/purge/erase the CKDS and PKDS.
- Define new/empty CKDS and PKDS.
- Change the ICSF parms for CKDS and PKDS.
- Stop ICSF.
- Start ICSF. (You'll get a message that MKs do not match.)
- Load new dummy symmetric master key.
- Initialize the CKDS.
- Load the same dummy symmetric master key AGAIN.
- Change master key.
- Load new dummy asymmetric master key.
- Load new dummy asymmetric master key.
- (With PKA, registers are automatically changed.)

Clear out the CEX2C and the key data sets!

It is important to clear out all your master and data encrypting keys before leaving the DR site.

Agenda

Verizon Wireless' business and database environments.

Evaluation Process

Secure vs. clear key.

Implementation process.

Disaster Recovery.

Auditing.

Conclusion.

Supplemental Material: ICSF, Key Labels and EDITPROCs.

Auditing

- We needed to report on any access to DB2 tables containing sensitive customer data. In addition to insert/update/delete activity, we needed visibility into who is reading the tables.
- We altered each our sensitive DB2 tables to "audit all." (Rebinds required.)
- We then activated the actual trace which was done from the DB2 command line.
  - *—STA TRACE(AUDIT) CLASS(1,2,3,4,5,6) DEST(SMF) XCONNID(CICS)*
- The trace resulted in a ~8% overall SMF increase.
- IFCID 145 is all DML access; 144 is read activity; 143 would show any structure changes.

Auditing (Con't)

Example DB2PM report control card

```
...
//SYSIN    DD *
  GLOBAL  PRESORTED(ACCEPT)
   TIMEZONE  (+08:00)
RECTRACE
   TRACE
     FROM (06/08/08,00:00:01)
     TO   (06/08/08,23:59:00)
     LEVEL  (SHORT)
     SORTBY (TIMESTAMP)
     INCLUDE (IFCID(143,144,145))
  EXEC
```

Example DB2PM Report Output:

```
145 AUDIT DML     STATEMENT     LOCATION NAME: xxxxxxxx
    PKG COLLCT ID: FILEDB2
    PROGRAM NAME : CWSQLPRO
    TIME: X'17EACB071E2A4AF7'
    TYPE: DELETE    STMT#:
    HOST OPTIONS    X'0400000000000000'
    SQL TEXT: DELETE FROM XXX.tablename WHERE BL_TYP_CD = ? AND INVOICE_NO = ?

145 AUDIT DML   STATEMENT     LOCATION NAME: xxxxxxxx
    PKG COLLCT ID: DSNESPCS
    PROGRAM NAME : DSNESM68
    TIME: X'149EEA901A79FE48'
    TYPE: SELECT - QUERY            STMT#:        0
    HOST OPTIONS    X'0400000000000000'
    SQL TEXT: SELECT * FROM XXX.tablename                         ISOLATION: CS.
```

Auditing (Con’t)

The audit gap

- When a vendor unload is executed against the DB2 VSAM data sets instead of through DB2, the IBM audit record has no knowledge of data access. However, the vendor utility “history” table will contain the date and time of the utility with the relevant utility id. The utility activity at run time is kept in another “in-flight” table. But the records are deleted upon completion of the utility.

Closing the Gap

- A DB2 trigger is deployed on the “in-flight” table that checks against the list of sensitive tablespaces. If it is one of our audited objects, the after trigger executes to insert this information into the DBA version of the in-flight table.

- CREATE TRIGGER
- xxxxx.trigger name
- AFTER
- INSERT
- ON xxxxx.DBA_UTILITY_INFLIGHT
- REFERENCING
- NEW AS N
- FOR EACH ROW
- MODE DB2SQL
- WHEN (N.NAME2 IN ('TS1', 'TS2', 'TS3', 'TS4','TS5')) BEGIN
- ATOMIC INSERT INTO xxxxx. DBA_UTILITY_INFLIGHT (UTILID, NAME1, NAME2, KIND,
- PARTITION, UTILNAME, SHRLEVEL, STATUS, XCOUNT, DDNAME,
- BLOCKS, ORIG_STATUS, EXTRBA, STATE) VALUES (N.UTILID, N.NAME1,
- N.NAME2, N.KIND, N.PARTITION, N.UTILNAME, N.SHRLEVEL,
- N.STATUS, N.XCOUNT, N.DDNAME, N.BLOCKS, N.ORIG_STATUS, N.EXTRBA,
- N.STATE) ; END

- In DBA_UTILITY_INFLIGHT, the record will not be deleted and so the audit trail is left in tact. A separate query of this table will yield all vendor unload activity.

Agenda

Verizon Wireless' business and database environments.

Evaluation Process

Secure vs. clear key.

Implementation process.

Disaster Recovery.

Auditing.

Conclusion.

Supplemental Material: ICSF, Key Labels and EDITPROCs.

Conclusion

- All of our DB2 for z/OS database are successfully encrypted and we are thrilled with the results.
- We have not experienced performance degradation.

Successful implementation requires thorough understanding of encryption concepts along with the collaboration of many teams that may not have worked together before.

References/Acknowledgement

- *Ernie Mancill*—Data Management Technical Specialist
- E. H. Nachtigall—CISSP; CISA zSeries and Competitive Cryptography Certified I/T Consultant
- Greg Boyd—System z Crypto and eBusiness Security
- *Mary Petras*—Lab Advocate
- The IBM ATS team
- *IBM ICSF manuals*

Questions & Answers

Agenda

Verizon Wireless' business and database environments.

Evaluation Process

Secure vs. clear key.

Implementation process.

Disaster Recovery.

Auditing.

Conclusion.

Supplemental Material: ICSF, Key Labels and EDITPROCs.

Supplemental Material: ICSF, Key Labels and EDITPROCS

- The following slides are a more detailed look at creating a key label and EDITPROC via the ICSF and the IBM Encryption Tool for IMS and DB2.

ICSF Preliminary Actions

- Ensure all the RACF authorities are in place. Otherwise, as you submit the ICSF tasks they will partially complete causing you some headaches.
- Validate the CKDS and PKDS data sets. There should be a node to distinguish DEV/TEST data sets from production. You'll need multiple data sets for key rotation. For example:
  - SYSMVS.ICSF.datacntr.plexname.CSFCKDS1
  - SYSMVS.ICSF.datacntr.plexname.CSFPKDS1
  - SYSMVS.ICSF.datacntr.plexname.CSFCKDS2
  - SYSMVS.ICSF.datacntr.plexname.CSFPKDS2
- Make sure the key data sets are empty. If necessary, you can use a tool like Fileaid to edit the VSAM data sets.
- Consider have a set of key data sets at the LPAR or application level for when you rotate master keys.
- Use ICSF Co-processor Management Verify the CEX2 coprocessors are in place.
- If you have multiple LPARS (parallel sysplex with DB2 data sharing), select the first LPAR. The steps are very different for the subsequent LPARs in the sysplex.

ICSF Steps: First LPAR in the SYSPLEX

- Perform Pass phrase master key and CKDS initialization.
  - Enter the pass phrase (16-64 characters). Be sure to use numbers, letters, and misspelled words.
  - Enter the CKDS and PKDS data sets in quotes.
  - Set Initialize the CKDS and PKDS? (Y/N) to Y.
  - Set Initialize new online coprocessors only? (Y/N) to N.
  - You should see a message like: *the master keys registers are being loaded. Initializing the key data sets*.
  - Then you will be prompted to **proceed with pass phrase initialization**. Hit <ENTER> to proceed.
  - Finally you should get an *Initialization Complete* message in the upper right hand corner.

ICSF Steps: First LPAR in the SYSPLEX (Con't)

- Select option #8 **Key Generator Utility Processes** to create the control cards.
- Select option #1 **Create - Create Key Generator Control Statements.**
- Enter the data set name (in quotes) for CSFIN <ENTER>.
- Enter the data set allocations for the (very small) CSFIN data set <ENTER>.
- Now create the key label:
  - Select option #1 **Maintain - Create ADD, UPDATE, or DELETE Control Statements.**
  - Populated the screen as follows: (next slide)

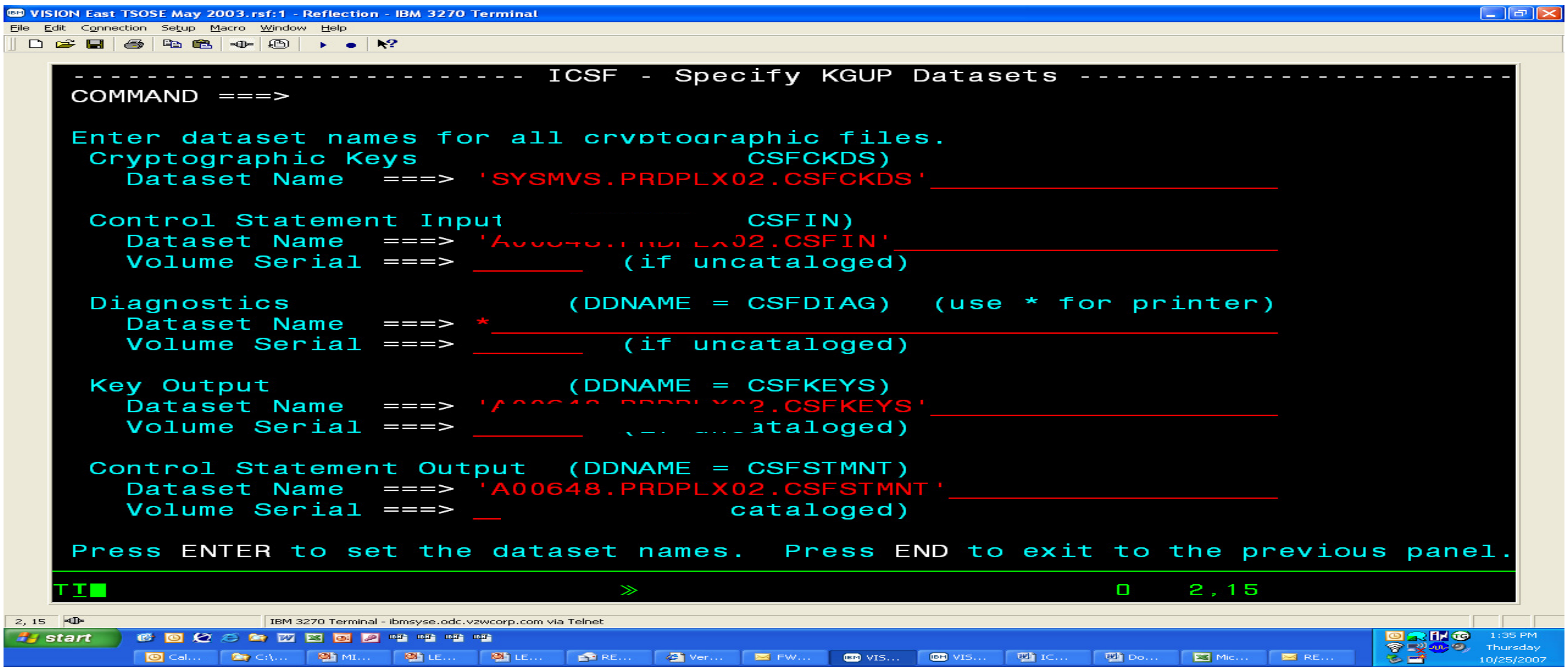
ICSF Steps: First LPAR in the SYSPLEX (Con't)

```
------------- ICSF - Create ADD, UPDATE, or DELETE Key Statement ------------
COMMAND ===>
Specify control statement information below

  Function ===> __ADD__     ADD, UPDATE, or DELETE
  Key Type ===> __CLRDES_   Outtype ===>              (Optional)
  Label ===> ___Whatever you want the security label name to be_________________
   Group Labels  ===> NO_   NO or YES
 or Range:
  Start ===> ______________________________________________________________
  End   ===> ______________________________________________________________

   Transport Key Label(s)
      ===> ______________________________________________________________
      ===> ______________________________________________________________
 or Clear Key            ===> NO_       NO or YES

  Control Vector ===> YES  NO or YES
  Length of Key  ===> _24  8, 16 or 24     For AES: 16, 24, or 32
  Key Values     ===>
   _______________ , _______________ , _______________ , _______________
  Comment Line   ===> _____________________________________________________
Press ENTER to create and store control statement
Press END   to exit to the previous panel without saving
```

- You should get a successful message after hitting <ENTER>.

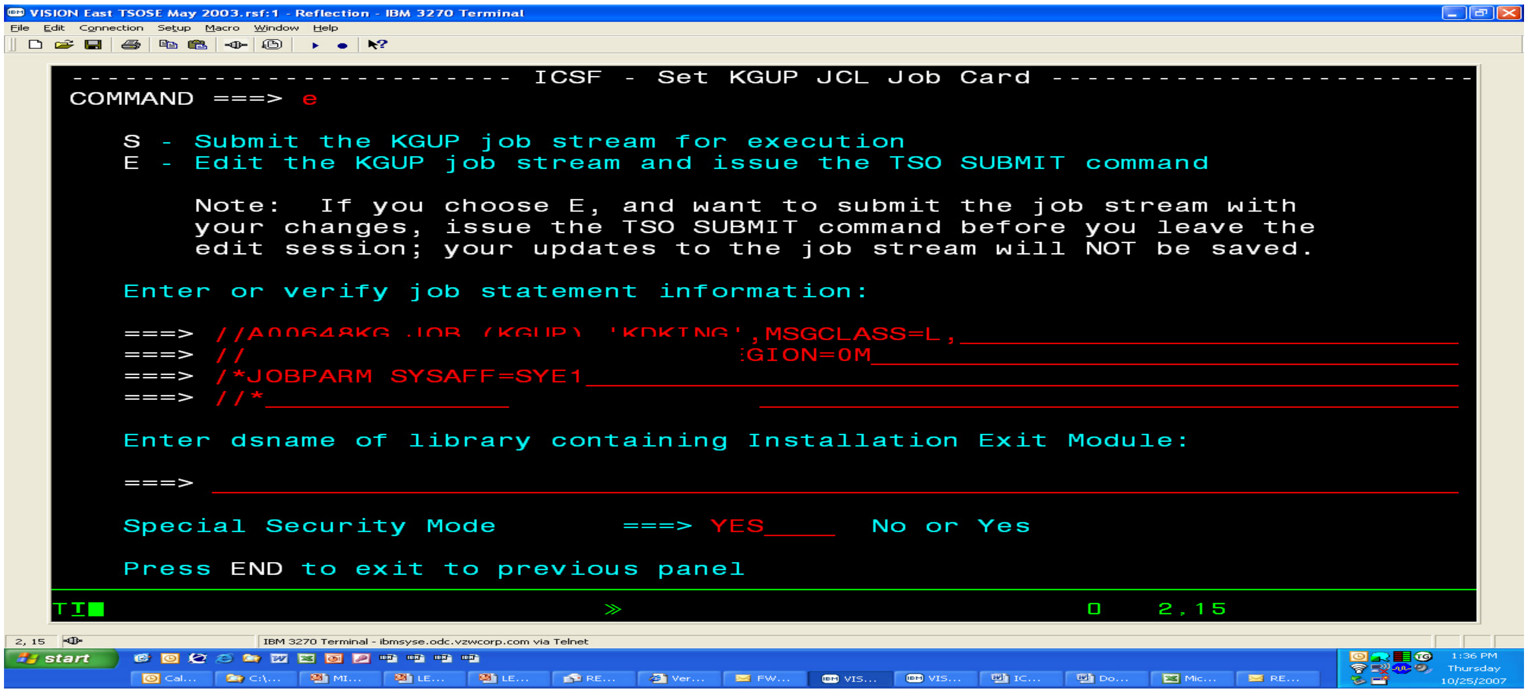
ICSF Steps: First LPAR in the SYSPLEX (Con't)

- PF3 twice
- Specify the Key Generation Data Sets



ICSF Steps: First LPAR in the SYSPLEX (Con't)

- Select option #3 Submit
- Now submit the job



ICSF Steps: First LPAR in the SYSPLEX (Con't)

- After submitting the job, go to SDSF and browse the job:

```
SDSF OUTPUT DISPLAY              JOB07701  DSID   101 LINE 0      COLUMNS 02- 81
 COMMAND INPUT ===>                                        SCROLL ===> CSR
******************************* TOP OF DATA ***************************
KEY GENERATION DIAGNOSTIC REPORT                                    DATE
ADD TYPE(CLRDES) LENGTH(24),
 LAB(Your label name will be here)
>>>CSFG0321 STATEMENT SUCCESSFULLY PROCESSED.
>>>CSFG0002 CRYPTOGRAPHIC KEY GENERATION - END OF JOB.  RETURN CODE = 0.
****************************** BOTTOM OF DATA *********************
```

ICSF Steps: First LPAR in the SYSPLEX (Con't)

- Select option #4 Refresh
- Refresh the in-storage CKDS.
- Enter the CKDS data set name in quotes and hit <ENTER>.
- Go back to coprocessor management (option #1) and select each coprocessor in order to verify the keys have been entered properly.

ICSF Steps: Subsequent LPARs in the SYSPLEX

Perform Pass phrase master key and CKDS initialization.

- Display the coprocessors from coprocessor management.
- Enter the same pass phrase used on the first LPAR.
- Enter the (same) CKDS and PKDS data sets in quotes.
- Set Initialize the CKDS and PKDS? (Y/N) to **N**.
- Set Initialize new online coprocessors only? (Y/N) to **N**.
- You should see a message regarding the crypto express coprocessors.
- Display the coprocessors from coprocessor management.
- Refresh the in-storage CKDS just like on the first LPAR.
- Finally you should get an *Initialization Complete* message in the upper right hand corner.

Thank you!

DB2 Encryption on z/OS plus a Real-life Success Story at Verizon Wireless

Ernest Mancill

IBM

mancill@us.ibm.com

Karl D. King

Verizon Wireless

karl.king@verizonwireless.com