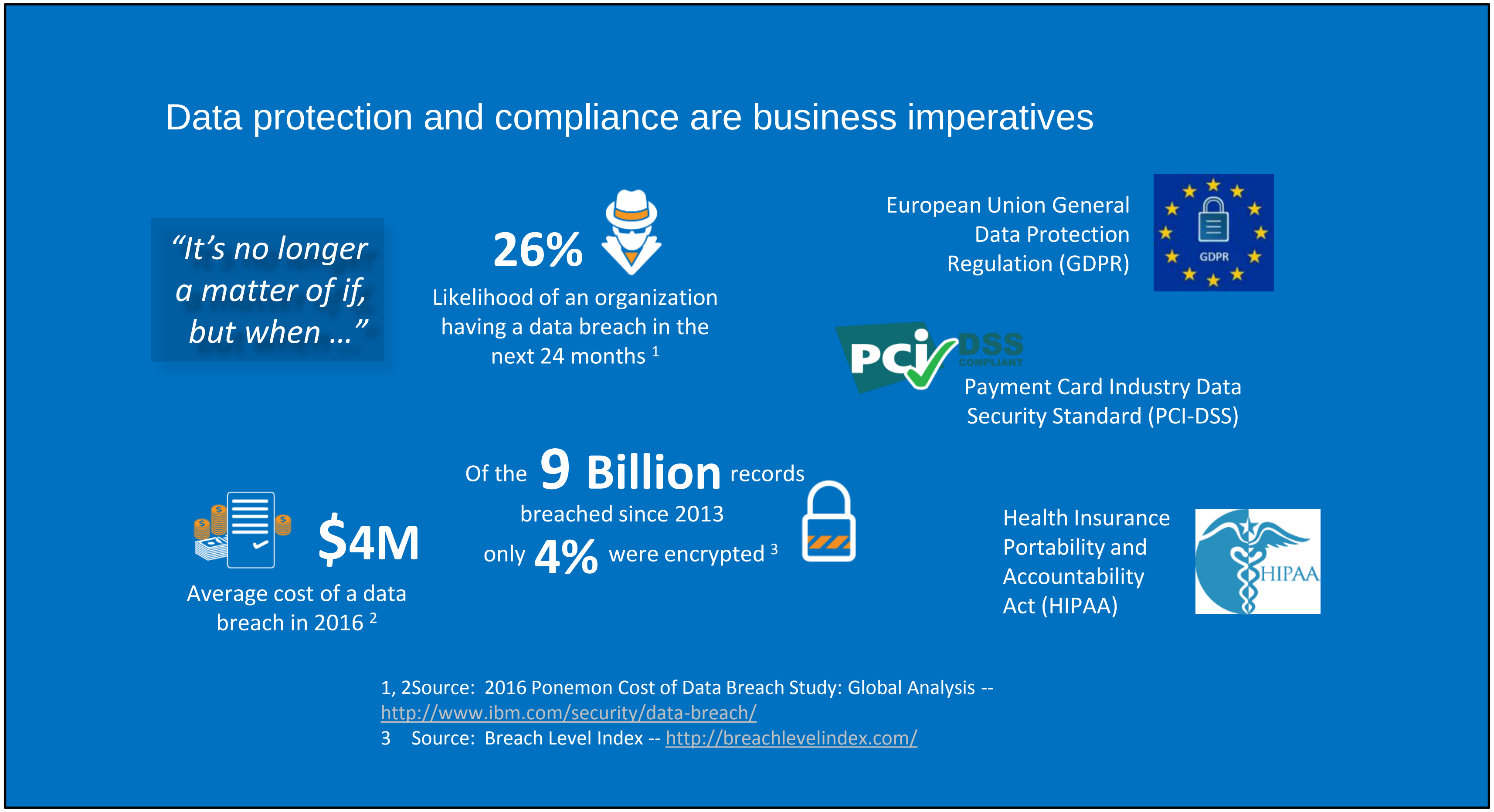
IMS Version 15

IMS 15 - Pervasive Encryption and IMS

Dennis Eichelberger
IMS Support Washington Systems Center
Deichel@us.ibm.com



Data protection and compliance are business imperatives

"It's no longer a matter of if, but when …"

26%
Likelihood of an organization having a data breach in the next 24 months [1]

European Union General Data Protection Regulation (GDPR)

Payment Card Industry Data Security Standard (PCI-DSS)

$4M
Average cost of a data breach in 2016 [2]

Of the **9 Billion** records breached since 2013 only **4%** were encrypted [3]

Health Insurance Portability and Accountability Act (HIPAA)

1, 2Source: 2016 Ponemon Cost of Data Breach Study: Global Analysis -- http://www.ibm.com/security/data-breach/
3 Source: Breach Level Index -- http://breachlevelindex.com/

“It’s not a matter of if, but when.”

- Regulatory Compliance
- Industry standards
- Customer satisfaction

- Many data breaches occur without immediate knowledge.

- 2016 – Average time to breach discovery - 191 days *

- Discovery of a breach may be *<u>much</u>* later.

There are No **“do overs”**

* Ponemon/IBM 2017 report

Pervasive Encryption with IBM z Systems

Enabled through tight platform integration

Network Encryption	Protect network traffic using standards based encryption from end to end, including encryption readiness technology[2] to ensure that z/OS systems meet approved encryption criteria
Data Set & File Encryption	▪ Protect Linux file systems and z/OS data sets[1] using policy controlled encryption that is transparent to applications and databases
Coupling Facility	Protect z/OS Coupling Facility[2] data end-to-end, using encryption that's transparent to applications
Secure Service Container	Secure deployment of software appliances including tamper protection during installation and runtime, restricted administrator access, and encryption of data and code in-flight and at-rest
Integrated Crypto Hardware	Hardware accelerated encryption on every core – CPACF PCIe Hardware Security Module (HSM) & Cryptographic Coprocessor – Crypto Express5S

And we're just getting started …

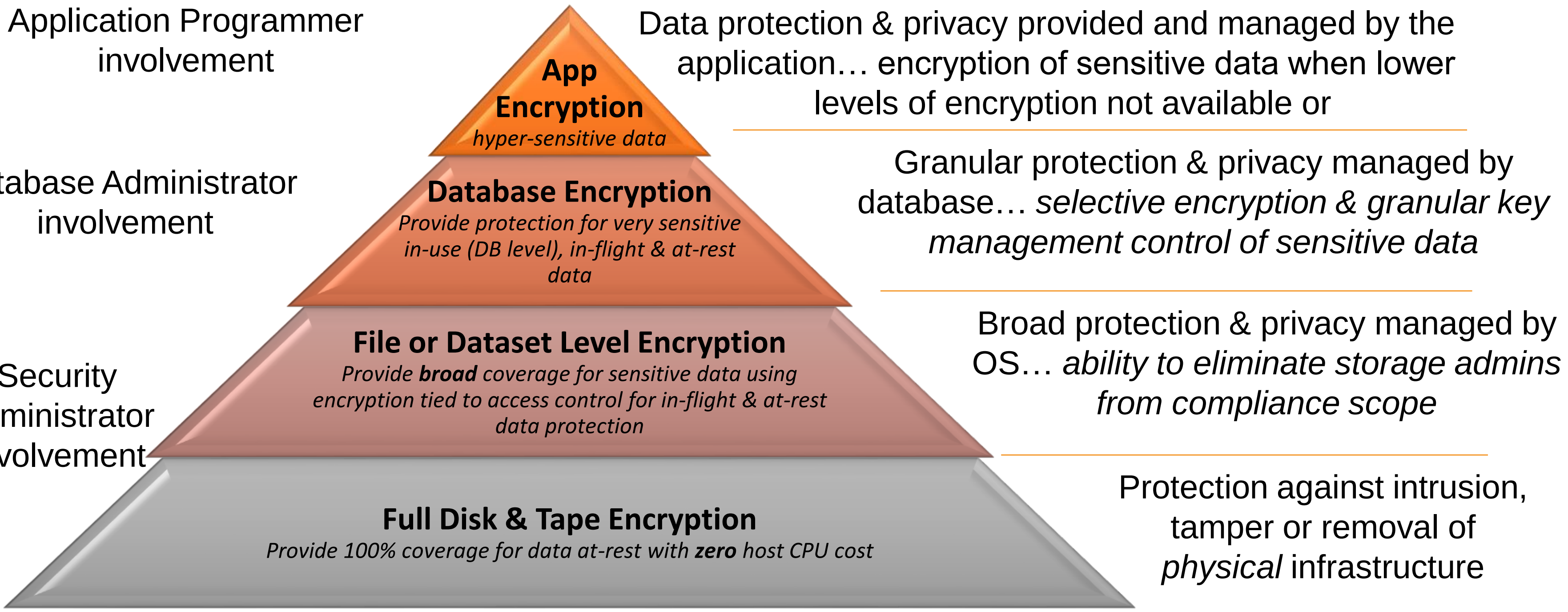
1 Statement of Direction* in the z/OS Announcement Letter (10/4/2016) - http://ibm.co/2ldwKoC
2 IBM z/OS Version 2 Release 3 Preview Announcement Letter (2/21/2017) - http://ibm.co/2l43ctN

* All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Topics

Levels of Encryption depend on where the data needs to be encrypted

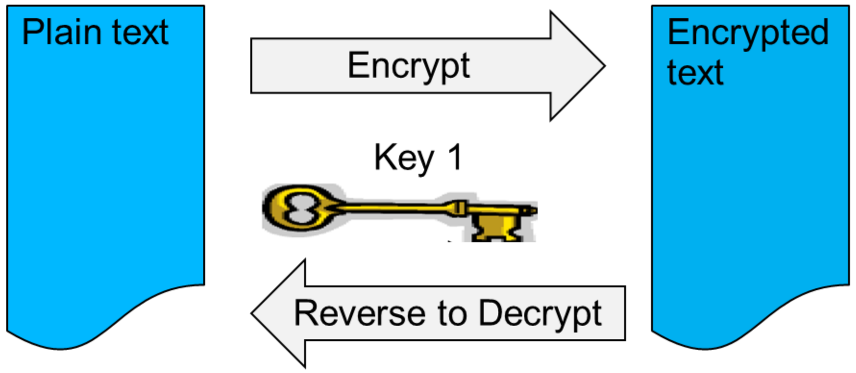
Implementation depends on resources and expected results


Application Programmer involvement
Database Administrator involvement
Security Administrator Involvement
App Encryption
hyper-sensitive data
Database Encryption
Provide protection for very sensitive in-use (DB level), in-flight & at-rest data
File or Dataset Level Encryption
Provide **broad** coverage for sensitive data using encryption tied to access control for in-flight & at-rest data protection
Full Disk & Tape Encryption
Provide 100% coverage for data at-rest with **zero** host CPU cost
Data protection & privacy provided and managed by the application… encryption of sensitive data when lower levels of encryption not available or
Granular protection & privacy managed by database… *selective encryption & granular key management control of sensitive data*
Broad protection & privacy managed by OS… *ability to eliminate storage admins from compliance scope*
Protection against intrusion, tamper or removal of *physical* infrastructure

Database Encryption

Encryption Algorithms

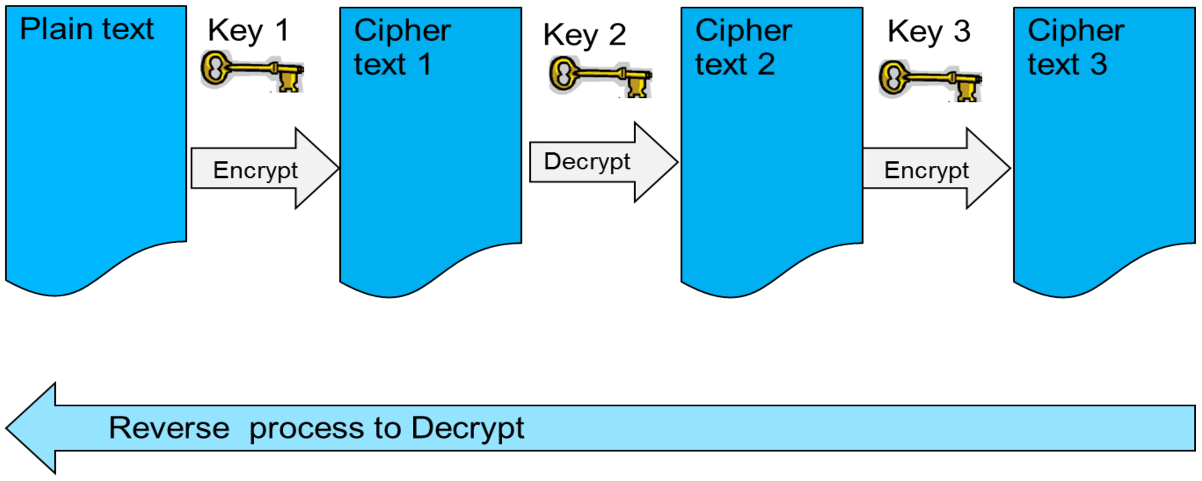
- DES (Data Encryption Standard)
  - 56-bit, viewed as weak and generally unacceptable today



Database Encryption

Encryption Algorithms

- TDES (Triple Data Encryption Standard)
  - 128-bit, accepted algorithm



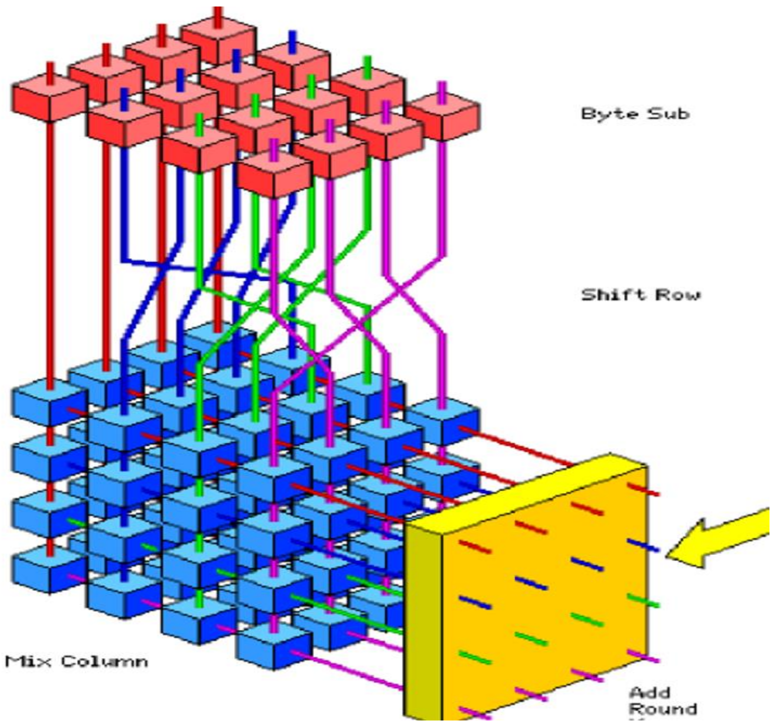
Note: same key can be used for each step for DES compatibility

Database Encryption

Encryption Algorithms

- (Advanced Encryption Standard)
  - 128-, 192- or 256- bit, commercially used algorithm

Rijndael Algorithm
- Block Cipher (16-byte blocks)
- 128, 192, 256-bit Key Length
- Multiple Rounds
- Four Steps per Round
  - Byte Substitution
  - Shift Row
  - Mix Column
  - Add Round Key



Database Encryption

Integrated Cryptographic Service Facility (ICSF)

- Provides: z/OS integrated software support for data encryption
- Operating System S/W API Interface to Cryptographic Hardware
  - CEX2/3C hardware feature for z114, z10 and z196
  - CEX4S hardware feature for z12BC and z12EC
  - CEX5S hardware feature for z13 (2x faster over CEX4S)
  - CEX6S hardware feature for z14 (5x plus faster over CEX5S)
- Enhanced Key Management for key creation and distribution
  - Public and private keys, Secure and clear keys, Master keys
- Created keys are stored/accessed in the Cryptographic Key Data Set (CKDS) with unique key label
- CKDS itself is secured via Security Access Facility

Database Encryption

What are Encryption Keys?

- Master Keys
  - Used to generate, encrypt, and store user keys into the CKDS (Cryptographic Key Data Set)
  - Loaded into the CEXnn hardware, and stored NO WHERE else
- User Keys (Data Encrypting Keys)
  - Generated via ICSF services
  - Stored inside the CKDS
  - Public or Private
  - Clear or Secure
  - Used by the IBM InfoSphere GDEz Encryption Tool along with encryption algorithm to convert user data to Ciphertext for Database Encryption

Database Encryption

Encryption Keys

- 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
  - CEXn hardware not required

- 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, 3, 4, 5, 6
  - APIs available via Integrated Cryptographic Support Facility (ICSF)
  - Can be used from Java on z/OS platform

Database Encryption

Encryption Keys

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

Database Encryption

Encryption Keys

- Clear Key vs. Secure Key Performance
- Protected Key
  - A Secure Key wrapped in encryption moved from the hardware
  - Reduces I/O type calls
  - Obscures Secure Key in memory as an encrypted object
  - Performance improvement over Secure Key
  - Greatest benefit for bulk processing against database

Application encryption

Data protection & privacy provided and managed by the application… encryption of sensitive data when lower levels of encryption not available or suitable

- Requires changes to applications to implement and maintain
- Highly granular
- Protect data right up to the point where it will be used
- Applications must be responsible for key management
- Appropriate for selective encryption of hyper-sensitive data

Application encryption

Db2 Built-In Functions

- Under application control – you encrypt the fields that need to be secure
  - ‘Password for Encryption’ is hashed to generate a unique key
  - Hint can be used as a prompt for remembering the key
  - Encrypted field must be defined as VARCHAR (since it will contain binary data once its encrypted)
  - The encrypted field will be longer (next multiple of 8 bytes + 24 bytes of MetaData + 32 bytes for optional hint field)
  - TDES Only!

```
Encrypt (StringDataToEncrypt, PasswordOrPhrase, PasswordHint)
Decrypt_Char(EncryptedData, PasswordOrPhrase
```

Application Encryption

Db2 Built-In Functions Example

```
CREATE TABLE EMPL
(EMPNO VARCHAR(64) FOR BIT DATA, EMPNAME CHAR(20),
CITY CHAR(20), SALARY DECIMAL(9,2))
IN DSNDB04.RAMATEST ;

COMMIT;
SET ENCRYPTION PASSWORD = ‘PEEKAY’ WITH HINT ‘ROTTIE’; INSERT INTO EMPL(EMPNO,
EMPNAME, SALARY)
VALUES (ENCRYPT(‘123456’),’PAOLO BRUNI’,20000.00) ;

INSERT INTO EMPL(EMPNO, EMPNAME, SALARY) VALUES (ENCRYPT(‘123457’),’ERNIE
MANCILL’,20000.00) ;
```

Bypasses Separation of Duty
- DBA or Programmer, maybe user, must know password
- Encryption Algorithm is limited –TDES only

From Redbook SG24-7959, Security Functions of IBM Db2 10 for z/OS

DATABASE ENCRYPTION

Database Encryption

Provide protection for very sensitive in-use (DB level), in-flight & at-rest data

Granular protection & privacy managed by database processes… *selective encryption & granular key management control of sensitive data*

- Encrypts sensitive data at the Db2 row and column levels and IMS segment level
- Transparent to applications
- Separation of Duties (SOD) and granular access control
- Protects Data-In-Use within memory buffers
- Clear text data cannot be accessed outside DBMS access methods
- Persists the encrypted sensitive data in logs, image copy data sets, DASD volume backups
- Utilizes IBM z Systems integrated cryptographic hardware

Database Encryption

IBM GDEz Data Encryption for Db2 and IMS Databases

- A Single tool for both Db2 and IMS
- Data Encryption addresses the increased demand for data privacy and security
- Performs encryption and decryption through the use of exit routines.
- Leverages the System z®, zSeries®, and S/390® Crypto Hardware to encrypt data
- Protects sensitive data that can reside on various storage media
  - Db2 and IMS databases
  - Image copy datasets
  - DASD volume backups
- Data is protected in DBMS buffers
- Does not bypass or preclude other encryption processes

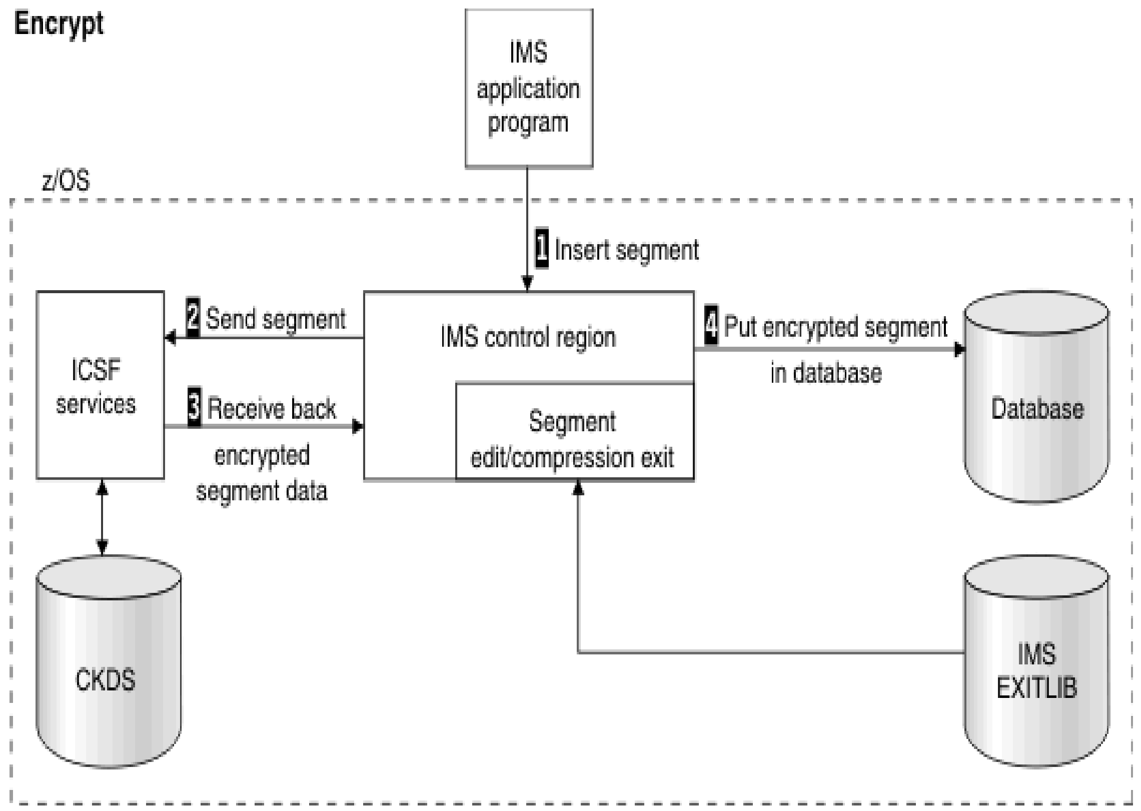
Database Encryption

Features and Benefits

- Enables compliance with privacy and security regulations
- Customization to Db2 column level and at the IMS segment level
- Straightforward implementation using defined key labels
- Ensures data privacy by encrypting and decrypting data
- Requires no changes to your applications
- Conforms to the existing z/OS and OS/390® security model
- Provides an ISPF front end to create and customize encryption exit routines
- Enables you to leverage the power of Storage Area Networks (SANs) safely while complying with privacy and security regulations
- Uses the following standard encryption algorithms:
  - ANSI Data Encryption Algorithm (DES), known as the Data Encryption Standard (DES)
  - Triple Data Encryption Algorithm (TDES), known as the Triple Data Encryption Standard (Triple DES)
  - Advanced Encryption Standard (AES) *

Database Encryption

IMS Encryption Flow

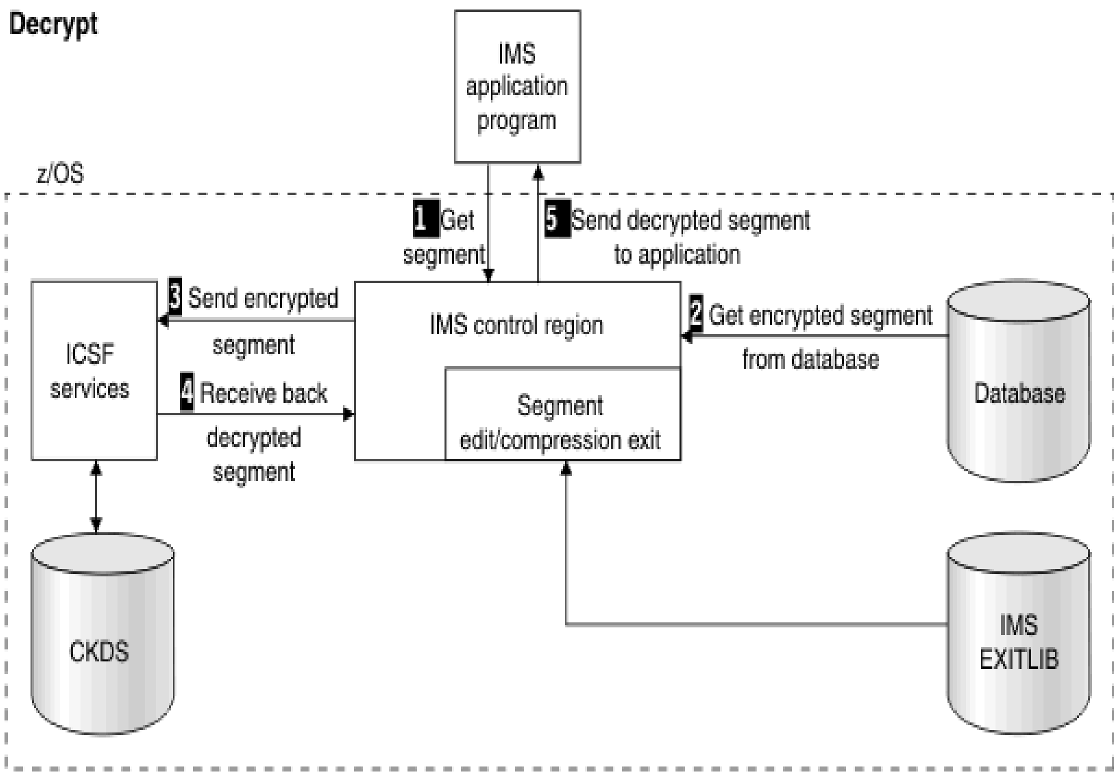


Encryption

1. IMS application program passes a segment REPL, ISRT, or LOAD request to the IMS control region. IMS uses the DBD to determine that a Segment Edit/Compression exit is required, so IMS loads the exit.
2. Exit invokes ICSF services, passing user-defined data encryption key label (provided by exit) and unencrypted segment.
3. When the segment has been successfully encrypted, the exit passes the segment back to IMS.
4. IMS then puts the encrypted segment into the database

Database Encryption

IMS Decryption Flow


Decrypt
IMS application program
z/OS
1 Get segment
5 Send decrypted segment to application
ICSF services
3 Send encrypted segment
4 Receive back decrypted segment
IMS control region
Segment edit/compression exit
2 Get encrypted segment from database
Database
CKDS
IMS EXITLIB

Decryption

1. IMS application program passes segment GET request to IMS control region. IMS determines, from DBD, that a Segment Edit/Compression exit is required, so IMS loads the exit.

2. IMS retrieves encrypted segment from the database.

3. IMS then calls the exit and passes it the encrypted segment. The exit invokes ICSF services, which passes the user-defined data encryption key label (provided by exit) and the encrypted segment.

4. When the segment has been successfully decrypted, the exit passes the segment back to IMS.

5. IMS passes the decrypted segment back to the application.

Database Encryption

IMS RESTRICTIONS

- An IMS segment is only associated with a single Segment Edit / Compression / encryption exit
  - An encryption exit may be associated with multiple segments
- An IMS segment with an existing segment exit should use a driver that ensures the correct call order e.g. Compress <u>before</u> Encryption
- HIDAM indexes cannot be encrypted using database encryption
  - IMS restriction

Database Encryption

IMS CONSIDERATIONS

- ICSF initialize with CHECKAUTH = NO
  - These options set to YES invoke extended path length

- A separate segment Edit / Compression exit needs to be built for each separate cryptographic key label

- A single key label may be used for multiple segments

- APF authorize the dataset containing the segment edit routines

- IMS loads the segment Edit / Compression routines below the 16M line
  - Be aware of storage use

- IMS databases allocated using OSAM dataset currently are encrypted ONLY by Guardium

Data Set Encryption

- Enabled by policy
- Transparent to applications
- Tied to access control
- Uses protected encryption keys managed by the host

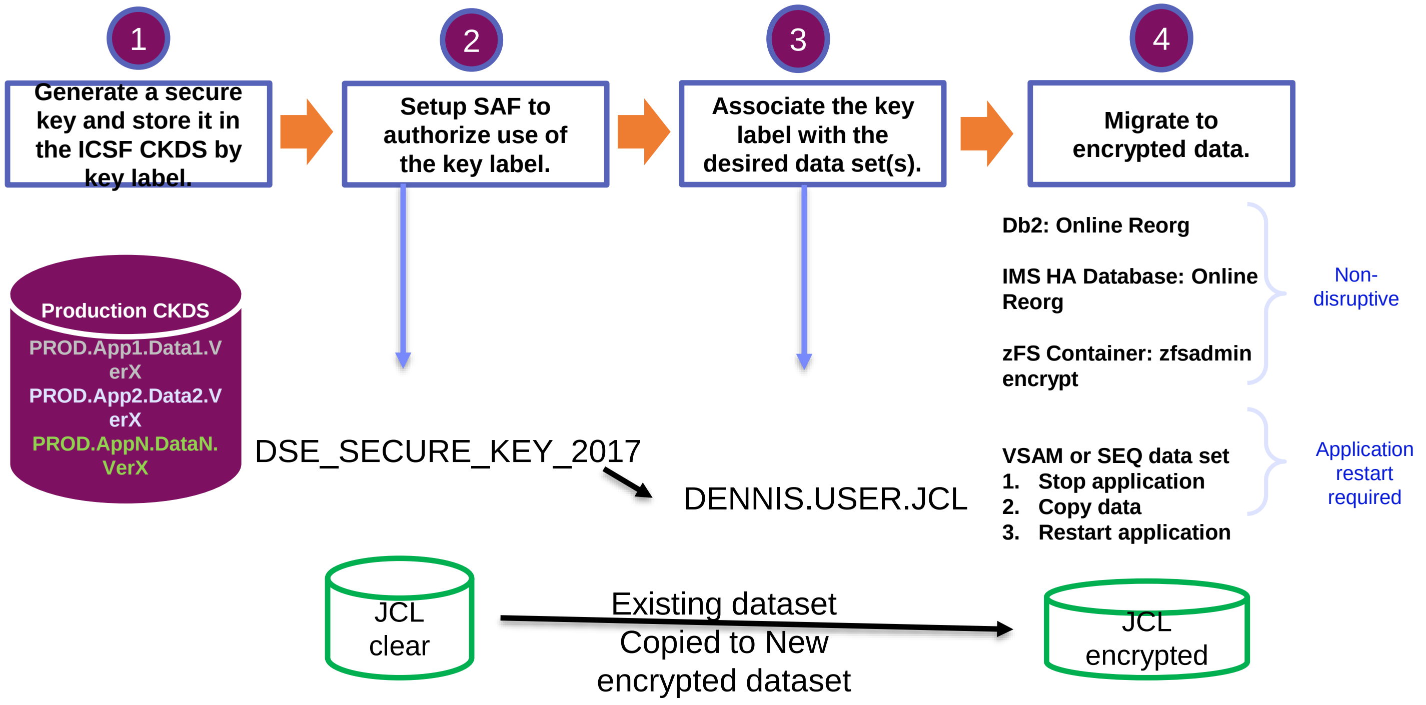
File or Data Set Level Encryption

*Provide **broad** coverage for sensitive data using encryption tied to access control for in-flight & at-rest data protection*

Broad protection & privacy managed by OS… *ability to eliminate storage admins from compliance scope*

- Broadly encrypt data at rest
- Covers VSAM, Db2, IMS, Middleware, Logs, Batch, & ISV solutions[1]
- Encrypt in bulk for low-overhead
- Utilizes IBM z Systems integrated cryptographic hardware

z/OS Data Set Encryption:
No Application Changes Required


1
Generate a secure key and store it in the ICSF CKDS by key label.
2
Setup SAF to authorize use of the key label.
3
Associate the key label with the desired data set(s).
4
Migrate to encrypted data.
Production CKDS
PROD.App1.Data1.VerX
PROD.App2.Data2.VerX
PROD.AppN.DataN.VerX
DSE_SECURE_KEY_2017
DENNIS.USER.JCL
Db2: Online Reorg
IMS HA Database: Online Reorg
zFS Container: zfsadmin encrypt
Non-disruptive
VSAM or SEQ data set
1. Stop application
2. Copy data
3. Restart application
Application restart required
JCL clear
Existing dataset Copied to New encrypted dataset
JCL encrypted

z/OS Data Set Encryption: Encryption By Policy

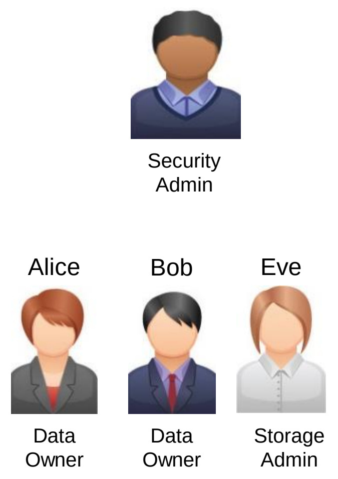
A data set is defined as ‘encrypted’ when a **key label** is supplied either on or prior to allocation of a ***new*** sequential or VSAM extended format data set.

The preferred method of enabling data set encryption is to specify a key label in the DFP segment of the RACF Data Set profile.

```
ALTDSD “PROJECTA.DATA.*’ UACC(NONE)
DFP(RESOWNER(ALICE) DATAKEY(key-label))
```

In the following example, Alice can read and update the data set. Bob can read the data set. Eve can read, update, delete, rename, move, or scratch the data set. But what content will they see?

```
PERMIT “PROJECTA.DATA.*” ID(ALICE)
ACCESS(UPDATE)

PERMIT “PROJECTA.DATA.*” ID(BOB) ACCESS(READ)

PERMIT “PROJECTA.DATA.*” ID(EVE) ACCESS(ALTER)
```



z/OS Data Set Encryption: Viewing the Content

Any user that needs access to the data set content in the clear must have access to the key label.

```
RDEFINE CSFKEYS * UACC(NONE)

RDEFINE CSFKEYS key-label UACC(NONE)

PERMIT key-label CLASS(CSFKEYS) ID(ALICE)
ACCESS(READ)

PERMIT key-label CLASS(CSFKEYS) ID(BOB)
ACCESS(READ)WHEN(CRITERIA(SMS(DSENCRYPTION))

PERMIT key-label CLASS(CSFKEYS) ID(EVE)
ACCESS(NONE)
```

In this example, Alice and Bob have access to the key label. So, they can view the data set contents in the clear. Eve has no access to the key label. So, even though she has UPDATE authority to manage the data set, she cannot view its contents.

Hardware encryption

- Protects at the DASD subsystem level
- All or nothing encryption
- Only data at rest is encrypted
- Single encryption key for everything

- No application overhead
- Zero host CPU cost
- Prevents exposures on: Disk removal, Box removal, File removal

Full Disk & Tape Encryption

*Provide 100% coverage for data at-rest with **zero** host CPU cost*

Protection against intrusion, tamper or removal of *physical* infrastructure

Hardware Encryption

Full Disk & Tape Encryption

DS8000 Manages the Complexity for You

- Most of the activity discussed above is managed transparently for the user once encryption is enabled and configured

- Encryption-related functions can all be handled through the DS8000 GUI or CLI

Storage Encryption configuration is really about authentication and separation of duties – the actual encryption is already happening inside the drives

Hardware Encryption

Full Disk & Tape Encryption

Authentication Keys

The FDE drives leave manufacturing in the unlocked state, so that all the disk operations are enabled, by default. In order to protect the data (to limit the read/write access), you can activate the locking mechanism by establishing new *authentication key*.

This key wraps (encodes) the encryption key. Therefore, from that point forward, the authentication key must be provided to the drive so that the drive can unwrap its encryption key for encoding and decoding the data transfer. The authentication key must be stored outside of the drive, available only for authorized systems. When the drive is powered off, it forgets its state, so it becomes locked. Every access request is denied while it is in locked state.

The proper authentication key is needed to unlock the drive temporarily. If the authentication process succeeds, client data can be accessed without any limitations until the next power loss.

Note:
The authentication key must be stored in a *safe* and *reliable* place. Unauthorized access to the key undermines the FDE security. The authentication key must be available continuously in order to unlock the drive whenever it is necessary. Losing all the copies of the authentication key yields to permanent data loss.

Hardware Encryption

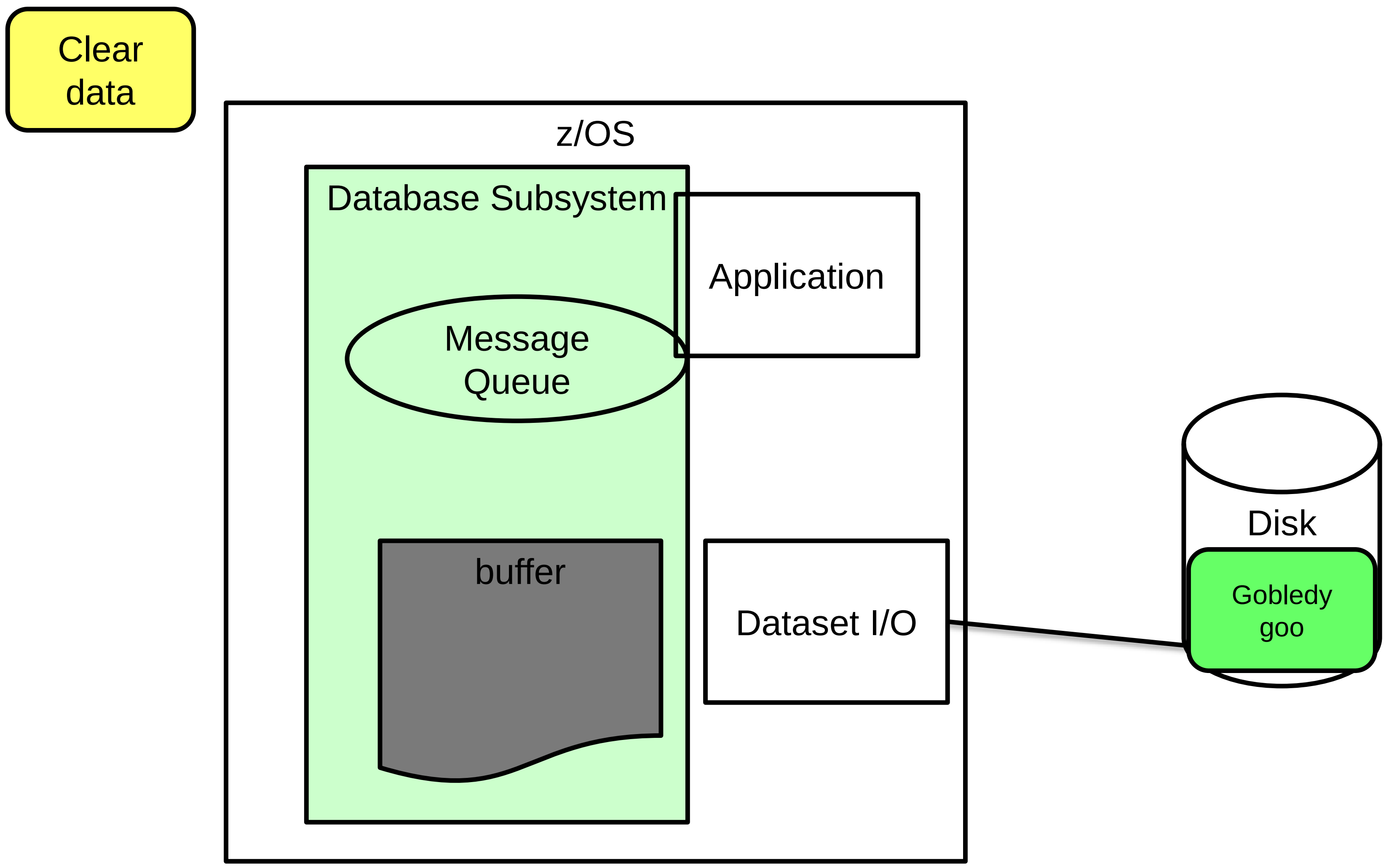
Full Disk & Tape Encryption

Protection against intrusion, tamper or removal of *physical* infrastructure

- Data is always encrypted on write to the drive and then decrypted on read
  - Data stored on the drive is encrypted
  - Customer data in flight is not encrypted
- Drives do the encryption at full data read/write rate
  - No impact to disk response times
  - Uses AES 256 bit encryption
- Supports cryptographic erasure data
  - Change of encryption keys
- Requires authentication with key server before access to data is granted
  - Key management is via IBM Security Key Lifecycle Manager (SKLM)
  - z/OS can also use IBM Security Key Lifecycle Manager (ISKLM)
  - Key exchange with key server is via 256 bit encryption
- Key attack methods addressed
  - Protection for disk removal (repair, replace or stolen)
  - Protection for disk subsystem removal (retired, replaced or stolen)

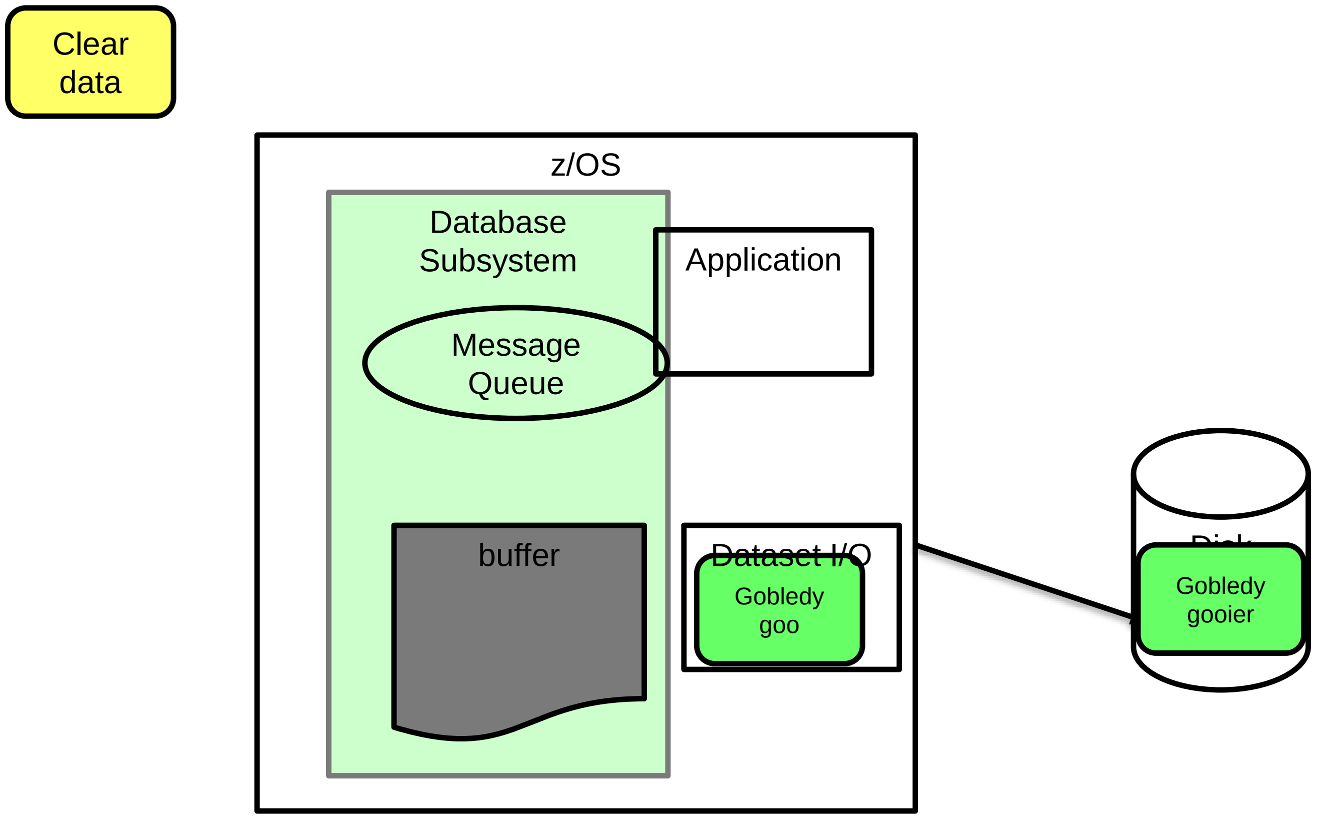
Database Encryption

Hardware Level



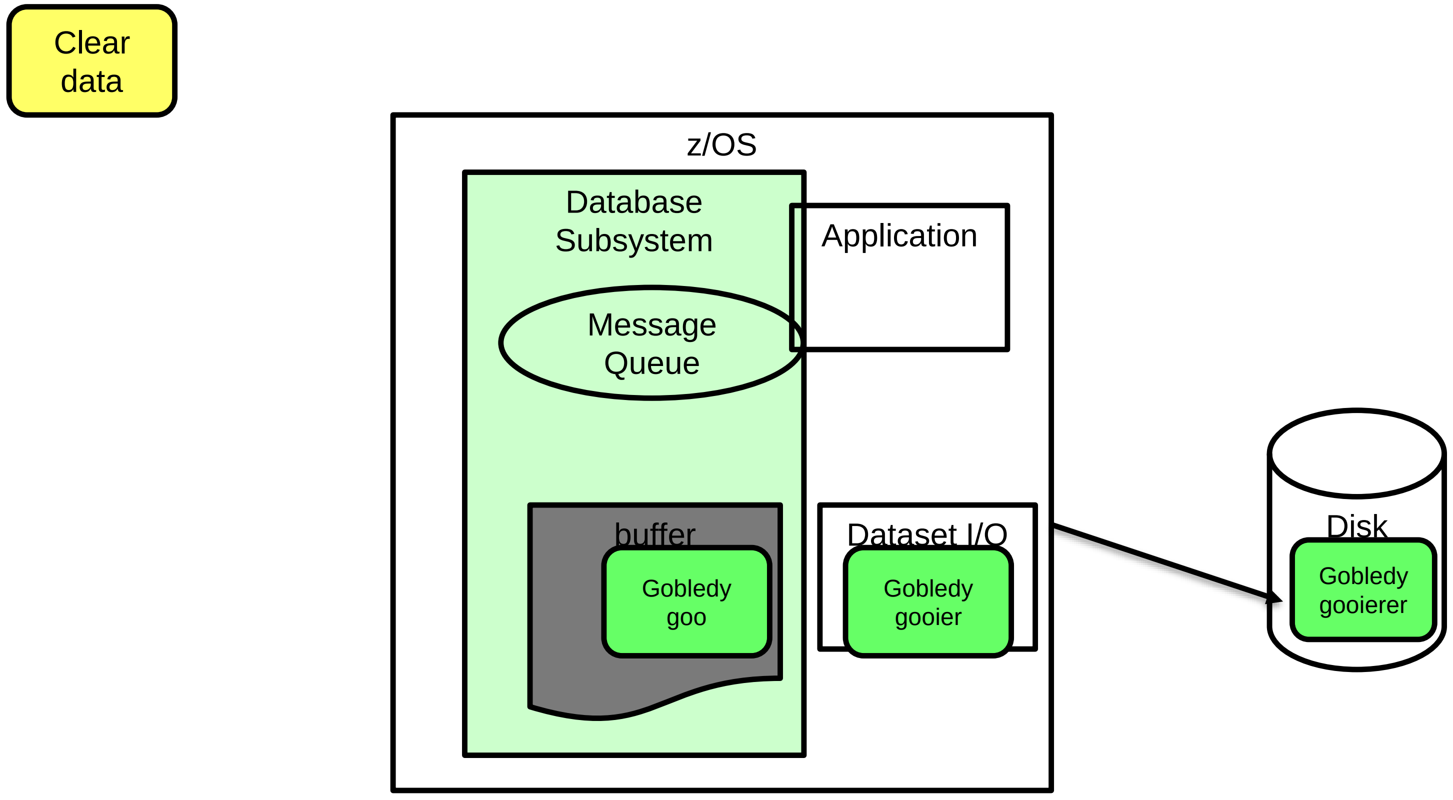
Database Encryption

Dataset Level



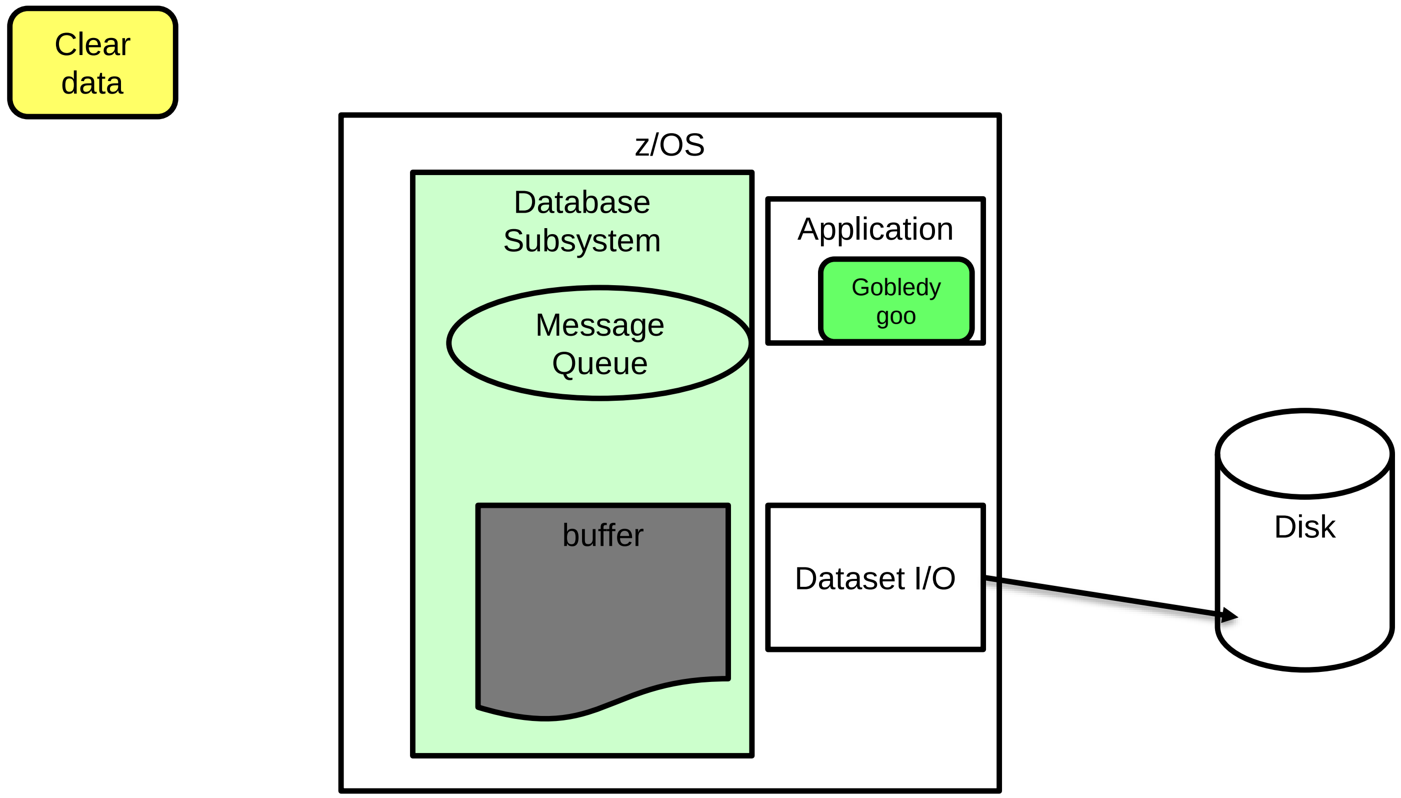
Database Encryption

Database Level



Database Encryption

Application Level



Encryption

Encrypted	Application	Database	Dataset	Disc
At rest	Yes	Yes	Yes	Yes
At Input / Output	Yes	Yes	Yes	No
In memory buffer	Yes	Yes	No	No
In Application program	Yes	No	No	No
Application Transparent	No	Yes	Yes	Yes
User transparent	Maybe	Yes	Yes	Yes
Maintenance Degree of Difficulty	High	Medium	Medium	Medium
Implementation Degree of Difficulty	High	Medium	Medium	Low

Encryption - IMS

IMS and Dataset Encryption

- Datasets used by IMS that are supported
- Datasets used by IMS that are *not* supported
- Datasets that exist will not be encrypted just by defining it so

Encryption - IMS

The following IMS V15 data sets supported in the DFSMS data set encryption:

Data Set Type	Notes	Address Space Userids Needing Key Label Access*
Database: VSAM (HALDB, non-HALDB)	Extended addressability attribute not supported for VSAM DBs.	CTL, DLI, batch jobs, utilities accessing DB
Database: DEDB	Added with APAR PI83756 PTF UI53418	CTL, DLI, batch jobs, utilities accessing DB
WADS (DFSWADSn)	VSAM Linear dataset. Must be allocated in Extended Format	CTL (including XRF alternate, FDBR regions), log archive utility, other utilities accessing OLDS, RSR transport manager
Online log data sets (DFSOLPnn, DFSOLSnn)		CTL (including XRF alternate, FDBR regions), log archive utility, other utilities accessing OLDS, RSR transport manager
Batch log data sets		CTL (including XRF alternate, FDBR regions), log archive utility, other utilities accessing OLDS, RSR transport manager
SLDS / RLDS		IMS batch jobs, utilities accessing batch logs, RSR transport manager

Encryption - IMS

The following IMS V15 data sets supported in the DFSMS data set encryption (cont):

Data Set Type	**Notes**	**Address Space Userids Needing Key Label Access***
Change Accum data sets	Change accumulation utility, DB recovery utilities	Change Accumulation data sets
Image copy data sets	Image copy utilities, DB recovery utilities	Image copy data sets
CQS SRDS		CQS
IMS Connect Recorder Trace		IMS Connect, utilities that process IMS recorder trace
BPE Trace data sets	Need to use RACF rules or DATACLAS with key label. Key label is not supported by BPE EXTTRACE statement.	Address spaces that use BPE, utilities that process BPE trace data (including IPCS TSO users)
z/OS log stream offload and staging data sets	Dependent on z/OS logger encryption support	z/OS logger address

Encryption - IMS

Encryption is explicitly *not* supported for the following IMS V15 data sets:

Data Set Type	**Notes**
Database: OSAM	EXCP / custom channel program; cannot be extended format
MSDB data sets	IBM has recommended MSDBs be converted to DEDBs for the last 10 IMS releases.
Queue manager data sets (LGMSG, SHMSG, QBLKS)	Uses OSAM
Restart data set (RDS)	Uses OSAM
All PDS / PDSE type data sets (PSBLIB, DBDLIB, ACBLIB, MODBLKS, FMTLIB, IMSTFMTx, IMSDALIB, program libraries, PROCLIB/configuration data sets, catalog directory, staging, BSDS)	DFSMS does not support PDS/PDSE encryption
Spool data sets	EXCP / custom channel program; cannot be extended format

Encryption IMS enablement

DFSMS data set encryption is established for a given data set when that data set is created and has a key label associated with it. Encrypted data sets must be extended format. Key labels can be specified for a data set by:

1. Creating RACF rules that associate a key label with a data set name pattern, via the DATAKEY parameter of the DFP segment.
2. Specifying a key label directly on JCL, dynamic allocation, TSO allocate, or IDCAMS DEFINE
3. Using a DATACLAS with a key label associated with it

Existing (already created) data sets that are not encrypted do not become encrypted just because their DATACLAS has a key label added to it, or because a RACF rule associates a key label with the data set. Existing data sets must be copied into a new extended format data set defined with a key label to become encrypted.

DFSMS Data Set Encryption

- Data sets are defined as encrypted by specifying a key label at the *creation* of a new data set:
  - SAF data set profile: Rules that associate a key label with a data set name pattern, via new **DATAKEY** parameter of the DFP RACF segment.
  - JCL, dynamic allocation, or TSO allocate (new **DSKEYLBL** parameter)
  - IDCAMS DEFINE (new **KEYLABEL** parameter)
  - SMS DATACLAS (new key label attribute)
- Application transparency: Data is encrypted/decrypted when accessed via supported access methods:
  - Data encryption/decryption occurs as data is written to or read from disk.
  - In-memory system or application data buffers remain in the clear.
  - Data remains encrypted during backup/recovery, migration/recall, and replication.
  - Access to key label is controlled through SAF permissions, in addition to traditional data set permissions.
- Programs accessing data sets using other access methods (Media Manager, direct channel programs) cannot access data sets encrypted by DFSMS without modification.

Encryption IMS enablement

Full Function VSAM Database Data Sets (non-HALDB, or not OLR-capable):

1. Preallocate the new database data sets with appropriate key labels
2. Take the database offline: e.g., /DBR DB, UPDATE DB STOP(ACCESS)
3. Run the appropriate unload utility (DFSURUL0 or DFSURGU0) or tool
4. Run the appropriate reload utility (DFSURRL0 or DFSURGL0) or tool to reload the database into the encrypted data sets

   Note: that if prefix resolution is required, the sort that is done will create temporary work data sets that are not encrypted. DFSMS does not support sort work data sets as extended format.

5. Rename the old data sets to a backup name
6. Rename the new data sets to the correct database data set names
7. Bring the database online: e.g., /STA DB, UPDATE DB START(ACCESS)

You may also be able to use a separate online reorg product to perform a database reorganization without taking the database offline. Consult your online reorg tool documentation for details.

Encryption IMS enablement

What to Expect if a the USERID of an Address Space is Not Authorized to a Key Label

In general OPEN will fail for the data set in question, so the IMS behavior should be no different than if any other type of OPEN failure occurs. You would expect to see an IEC161I (open failure) and a RACF authorization failure like:

IEC161I *data_set_name*,,VCATSHR

ICH408I USER(OMVSADM) GROUP(SYS1) NAME(OMVS) 872 *key_label*

CL(CSFKEYS)

INSUFFICIENT ACCESS AUTHORITY

ACCESS INTENT(READ) ACCESS ALLOWED(NONE)

followed by whatever error message the process deems appropriate for the open failure.

Encryption Implementation Considerations

- Application
  - Application logic determines which key to use for each field/column
  - Password is managed by the application
- Data Administrator - Data Encryption Tool
  - Sets up the EDITPROC and specifies the key to be used for the entire table – Db2
  - Sets up the COMPRTN and specifies the key for encrypted segments – IMS
  - Key must be defined to/managed by ICSF (stored in the CKDS)
- Security requirements
- Performance requirements
  - Single transaction vs bulk processes
- Application/production support
- Space considerations
- Crypto hardware available

Encryption Implementation Considerations

- Not insignificant
  - Multiple points of implementation
- May require multiple groups to coordinate
  - Security
  - Storage
  - System programmer
- Have a back up / back out plan

Encryption IMS enablement

Full Function VSAM Database Data Sets (HALDB, OLR capable):

Either use the offline process as described above for non-HALDB, or use HALDB Online Reorg to reorganize the unencrypted data set into an encrypted data set:

1. Preallocate the new OLR target database data sets with appropriate key labels.
2. Run IMS online reorg for one or more partitions. Repeat until all of the partitions are reorganized.

Encryption IMS enablement

Encrypting the Write Ahead Data Set (WADS)

The WADS can be encrypted across a restart of IMS or while IMS remains up. The steps for each procedure are outlined below.
Note that IMS can use up to 10 WADS, numbered 0-9

If different data set names are used for the encrypted and non-encrypted WADS, and MDA is not used for the WADS data set names (that is, there are DD statements in the control region JCL for the WADS), then IMS must be recycled to perform this migration.

This is because the online control region will have the WADS allocated and they will be unable to be deleted while IMS remains up.

Encryption IMS enablement

Encrypting the Write Ahead Data Set (WADS)

Allocating for Using Encryption

```
DEFINE CLUSTER -
(NAME(data set name) -
CONTROLINTERVALSIZE(4096) -
SHAREOPTIONS(3 3) -
CYL(200) -
DATACLAS(dataclas)-
STORCLAS(storclas)-
KEYLABEL(keylabel)-
LINEAR)
```

Keylabel is the label for the encryption key

Database Encryption

APPENDIX

Encryption IMS enablement

Encrypting the Write Ahead Data Set (WADS)

Procedure if IMS is to come down for migration:

1. Ensure that IMS comes down cleanly
2. Delete all WADS
3. Define all new WADS as extended format and with a key label.
    a) If different data set names are being used for the encrypted WADS, this is the point where the MDA members are to be updated or the DD statements in the control region JCL are to be changed.
4. Cold start IMS, specifying “FMT WA” or “FMT ALL” on the /NRESTART command.

Encryption IMS enablement

Encrypting the Write Ahead Data Set (WADS)

Procedure for keeping IMS up during the migration:

1. Issue /DIS OLDS to see which WADS are in use
2. Issue a “/STO WADS x” command for each WADS that’s not in use
3. Delete all of the WADS just stopped and define them as extended format and with a key label.
   a) If different data set names are being used for the encrypted WADS, this is the point where the MDA members are to be updated.
4. Issue “/STA WADS x” for each WADS just delete/defined.

   Note: WADS are formatted as part of /STA WADS processing
5. Issue /SWI WADS
6. Issue “/STO WADS x” for the WADS that were just switched from.
7. Note: If using dual WADS, this command will need to be issued for each WADS just switched from.
8. Delete/define the WADS just switched from as extended format and with a key label.
   a) If different data set names are being used for the encrypted WADS, this is the point where the MDA members are to be updated.
9. Issue “/STA WADS x” for the WADS just delete/defined.

Encryption IMS enablement

Online Log Data Sets (OLDS):
Use the following steps to convert to encrypted OLDS when you can shut down IMS and restart it across the conversion.

Note that you must use this approach if your OLDS are specified as DD statements in your IMS control region JCL: i.e., are not dynamically allocated:

1. Create a new set of encrypted OLDS with key labels – one OLDS for each existing OLDS in the set used by IMS..
2. Pre-format each new OLDS data sets. One way to do this is to copy an existing full OLDS into one of your new encrypted OLDS data sets. Then, use that copy to copy into all of your other new OLDS. This ensures that all blocks in the new OLDS are initialized.
3. If IMS is active, issue /CHE FREEZE and ensure that IMS comes down clean. Ensure that all log archive jobs complete successfully.
4. Remove the PRIOLD/SECOLD entries from DBRC using the DSPURX00 utility.
5. Rename the original OLDS data sets to a backup name, and rename the new encrypted OLDS to the original OLDS data set names.

Warm starting IMS (restart will use the SLDS).

6. You can delete the old OLDS data sets once you have completed the migration and have confirmed all is operating correctly.

Encryption IMS enablement

If IMS is active and you want to enable encryption for the OLDS without stopping and restarting IMS, you can either migrate one OLDS at a time, or, if you are using 50 or fewer OLDS, you can migrate from one “set” of unencrypted OLDS to a new “set” of encrypted OLDS.

Note that both of these sequences require the use of dynamic allocation for your OLDS data sets. You cannot migrate OLDS to be encrypted while IMS is active if there are DD statements in the control region JCL for the OLDS. This is because the procedure deletes or renames the old data sets and this will fail if the control region has an OLDS allocated.

Encryption IMS enablement

To migrate one OLDS at a time while IMS is active:

1. Create a new set of encrypted OLDS with key labels – one OLDS for each existing OLDS in the set used by IMS. The new OLDS should be the same attributes as the current OLDS. They must be allocated as extended format datasets.
2. Pre-format each new OLDS data sets:
3. One way to do this is to copy an existing full OLDS into one of your new encrypted OLDS data sets. Then, use that copy to copy into all of your other new OLDS. This ensures that all blocks in the new OLDS are initialized.
4. Either wait until a current OLDS fills and IMS switches to the next OLDS, or issue a /SWI OLDS command to force an immediate switch. Then, for the OLDS that you switched from:
   1. Wait until log archiving completes for the switched-from OLDS.
   2. Issue the **/STO OLDS nn** command for the switched-from OLDS.
   3. Delete the OLDS from DBRC by removing the PRIOLD/SECOLD entries using DSPURX00 utility or online command.
   4. Rename the old OLDS data set to another name. Do this for both the primary and secondary “switched-from” OLDS if you are using dual OLDS logging.
   5. Rename the new encrypted OLDS data set to the old OLDS data set name. Do this for both the primary and secondary OLDS if you are using dual OLDS logging.
   6. Issue a **/STA OLDS nn** command to start using the encrypted OLDS.
5. Repeat step 3 for each OLDS in turn until all OLDS have been converted to be encrypted.
6. You can delete the old OLDS data sets once you have completed the migration and have confirmed all is operating correctly.

Encryption IMS enablement

Batch Log Data Sets:
For new batch jobs, define the batch log data sets with a key label using one of the DFSMS methods for doing so. You cannot change a batch log to be encrypted while a batch job is executing.
SLDS/RLDS (IMS archived log data sets):
Update DBRC log archive skeletal JCL to specify either a key label via DSKEYLBL=, or a DATACLAS with an associated key label for the SLDS / RLDS data sets. Alternately, use RACF to associate a key label with your archive log data sets by data set name pattern. Newly-allocated SLDS and RLDS will then be encrypted.
Image Copy Data Sets:
Update DBRC image copy skeletal JCL to specify either a key label via DSKEYLBL=, or a DATACLAS with an associated key label for the image copy data sets. Alternately, use RACF to associate a key label with your image copy data sets by data set name pattern.
Change Accumulation Data Sets:
Update DBRC change accum skeletal JCL to specify either a key label via DSKEYLBL=, or a DATACLAS with an associated key label for the change accum data sets. Alternately, use RACF to associate a key label with your change accum data sets by data set name pattern.
Note that DFSMS does not support SORT work data sets being encrypted.
CQS SRDS Data Sets:
Use IDCAMS to define two new encrypted SRDSs the same size as or larger than the current SRDSs. For the purpose of this example, assume the original SRDSs were named SRDS1 and SRDS2, and the new SRDSs are named SRDS1.NEW and SRDS2.NEW.

Encryption IMS

OSAM dataset support direction

DBAs/System Programmers can enable z Systems hardware encryption for OSAM data without any changes to applications and without experiencing any outage failure.

IBM IMS

Continuous Customer **Engagement**

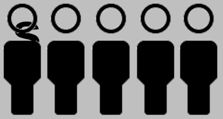
Design Thinking Based Continuous Delivery

Signature Client **Experience**

Use case driven
Ideation, Playbacks, demos
Deliver when ready

It all adds up to GOLD

- **All IMS Customer**



The **Global IMS Customer base** has the opportunity to become a feedback client

- **Z Client Feedback Program Agreement**

One-time enrollment process to be part of the Z Client Feedback program

- **IMS GOLD**

Bring all 5 Personas to the design table

-**create** with IBM in an **agile** setup

Hill-based benefits to each **persona**

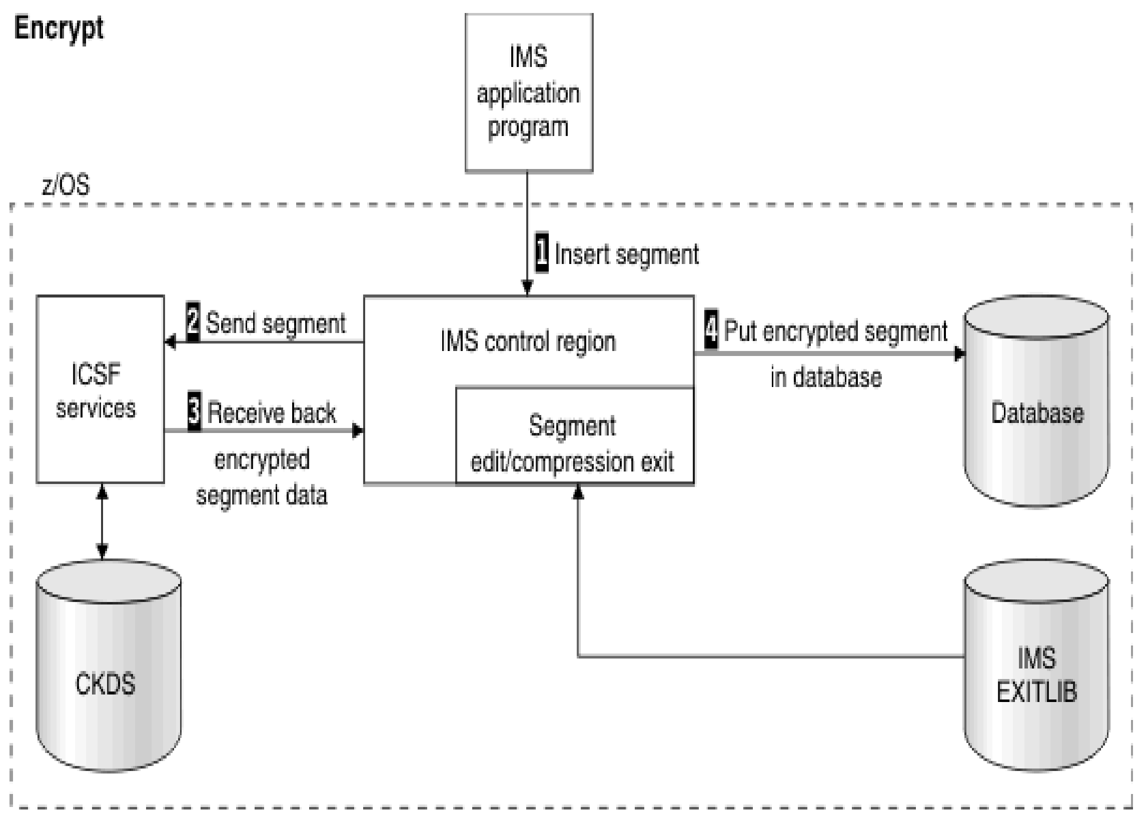
Provide **feedback** as features get designed and developed

Co-present with IBM

Exploit new functionality

Database Encryption

IMS Encryption Flow

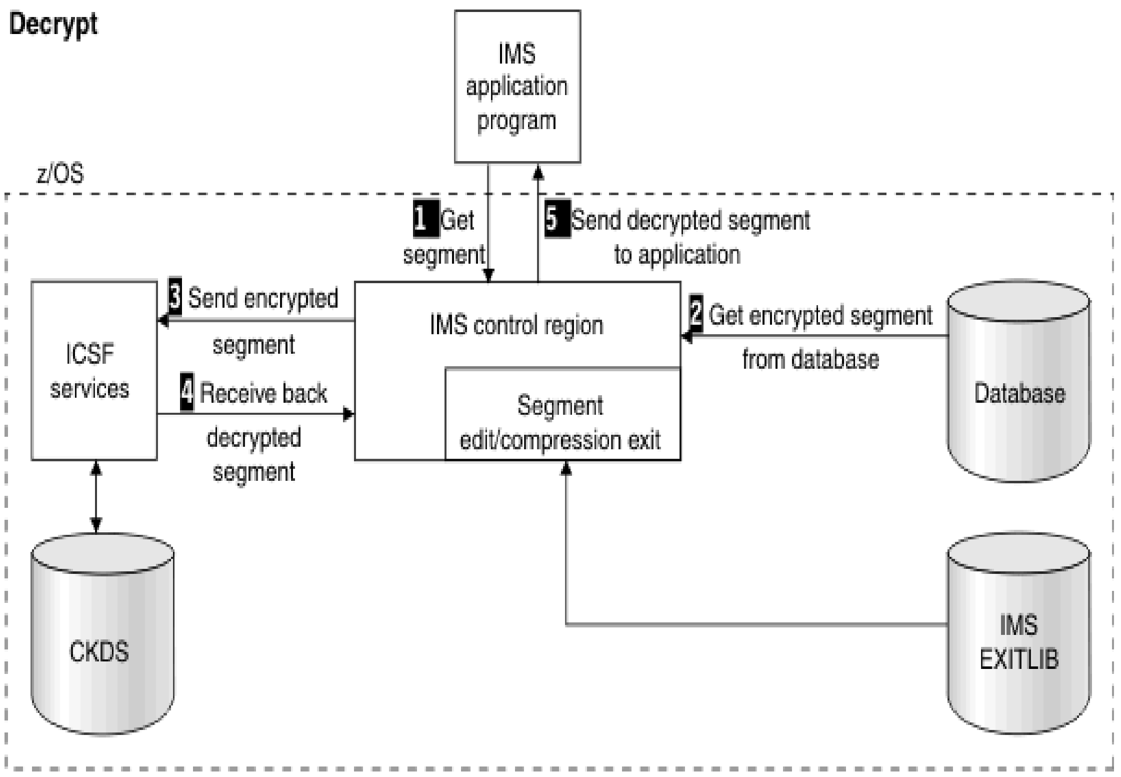


Encryption

1. IMS application program passes a segment REPL, ISRT, or LOAD request to the IMS control region. IMS uses the DBD to determine that a Segment Edit/Compression exit is required, so IMS loads the exit.
2. Exit invokes ICSF services, passing user-defined data encryption key label (provided by exit) and unencrypted segment.
3. When the segment has been successfully encrypted, the exit passes the segment back to IMS.
4. IMS then puts the encrypted segment into the database

Database Encryption

IMS Decryption Flow


Decrypt
IMS application program
z/OS
1 Get segment
5 Send decrypted segment to application
3 Send encrypted segment
IMS control region
2 Get encrypted segment from database
ICSF services
4 Receive back decrypted segment
Segment edit/compression exit
Database
CKDS
IMS EXITLIB

Decryption

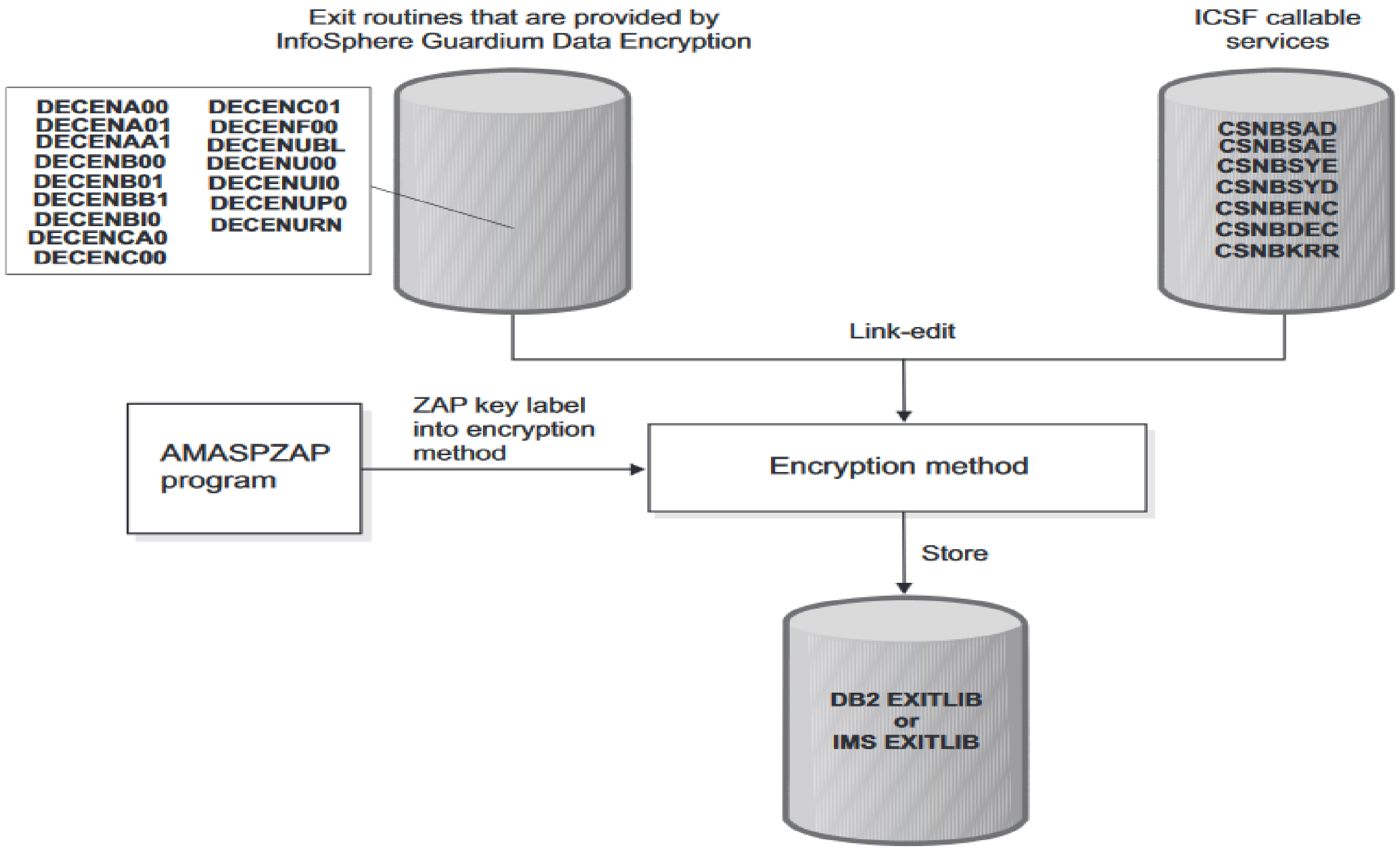
1. IMS application program passes segment GET request to IMS control region. IMS determines, from DBD, that a Segment Edit/Compression exit is required, so IMS loads the exit.
2. IMS retrieves encrypted segment from the database.
3. IMS then calls the exit and passes it the encrypted segment. The exit invokes ICSF services, which passes the user-defined data encryption key label (provided by exit) and the encrypted segment.
4. When the segment has been successfully decrypted, the exit passes the segment back to IMS.
5. IMS passes the decrypted segment back to the application.

DATA ENCRYPTION USING GDEZ FOR ENCRYPTION FOR IMS AND DB2

Database Encryption

IMS Supplied Exit Routines

IMS Exit Routine	Key Encryption
DECENA01	Clear Key
DECENAA1	Clear Key CPACF with Protected Key wrapping Batch CHECKAUTH recurring bypass
DECENB01	CPACF Protected Key
DECENBB1	CPACF Protected Key Batch CHECKAUTH recurring bypass
DECENC01	Secure Key


Exit routines that are provided by InfoSphere Guardium Data Encryption
DECENA00
DECENA01
DECENAA1
DECENB00
DECENB01
DECENBB1
DECENBI0
DECENCA0
DECENC00
DECENC01
DECENF00
DECENUBL
DECENU00
DECENUI0
DECENUP0
DECENURN
ICSF callable services
CSNBSAD
CSNBSAE
CSNBSYE
CSNBSYD
CSNBENC
CSNBDEC
CSNBKRR
Link-edit
AMASPZAP program
ZAP key label into encryption method
Encryption method
Store
DB2 EXITLIB
or
IMS EXITLIB

InfoSphere GDEz Data Encryption for Db2 and IMS Databases

Using the ISPF interface.

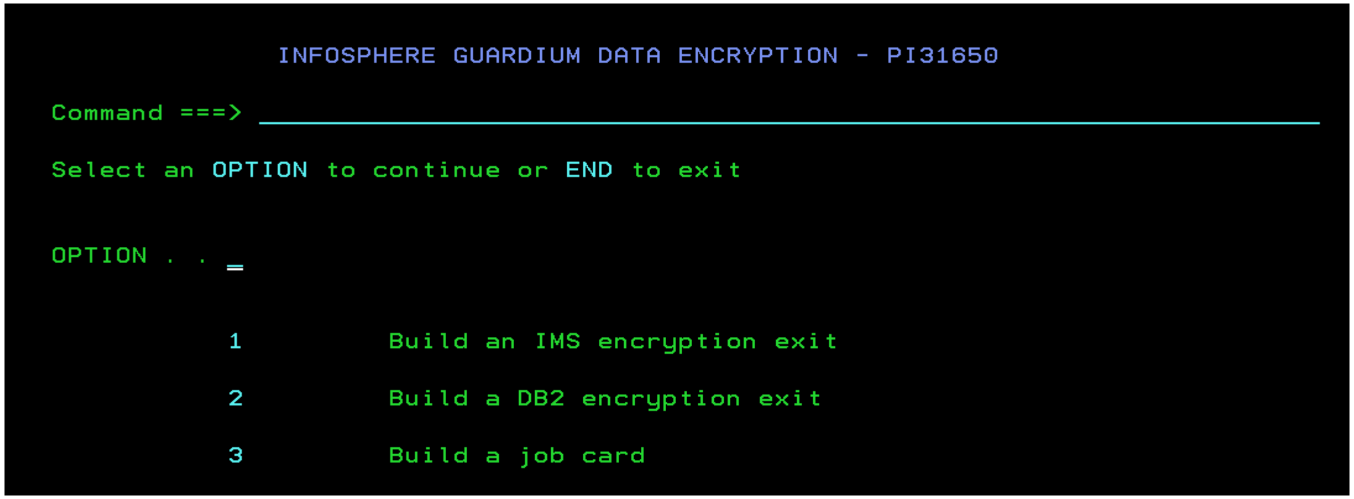
An ISPF dialog is available for you to create customized jobs for encrypting IMS database segments. The ISPF dialog creates customized JCL based on the sample jobs from the previous slide and edited with the information the user supplies.

Implementation with the Data Encryption Tool for IMS?

- Implementing IMS Encryption with the Data Encryption Tool
  - Generate Key using ICSF KGUP (Key Generation Update Program)
  - Prepare your exit using Data Encryption Tool providing ICSF Keylabel
  - Generate the DBD and ACB(s) to include the COMPRTN value
  - Unload target database
  - Activate the ACB to your IMS systems
    - Include the exit
    - Assure the exit is available in an APF library
  - LOAD the target database
  - /STA *database*
  - Encryption is now operational

Database Encryption

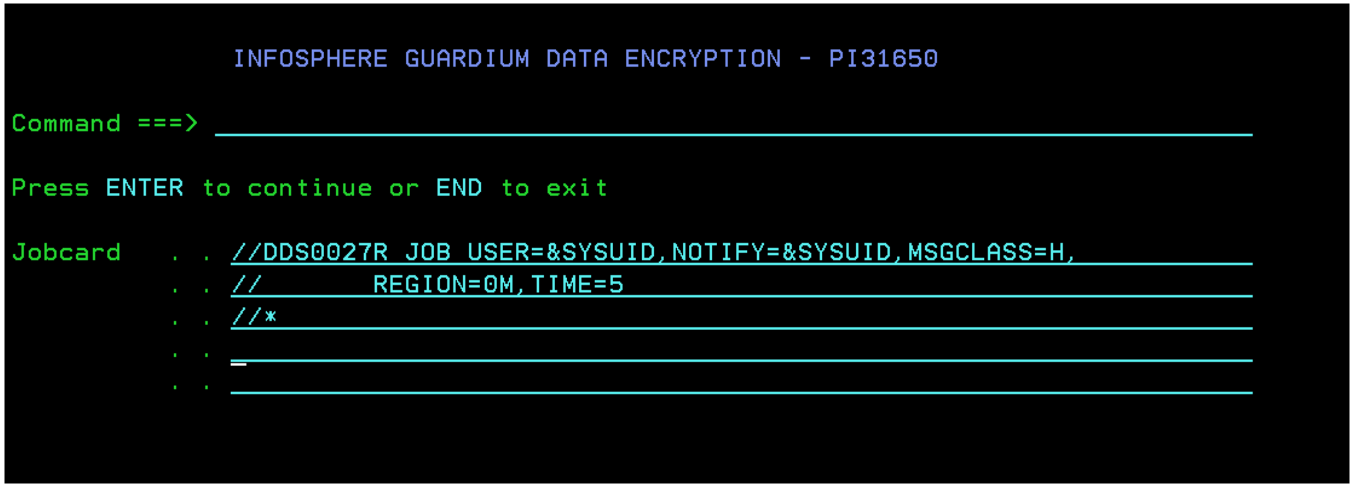
InfoSphere GDEz Data Encryption – ISPF Main Menu



```
                 INFOSPHERE GUARDIUM DATA ENCRYPTION - PI31650

Command ===> ___________________________________________________________________

Select an OPTION to continue or END to exit


OPTION . . _


           1          Build an IMS encryption exit

           2          Build a DB2 encryption exit

           3          Build a job card
```

Selection 3 for Jobcard creation

Database Encryption

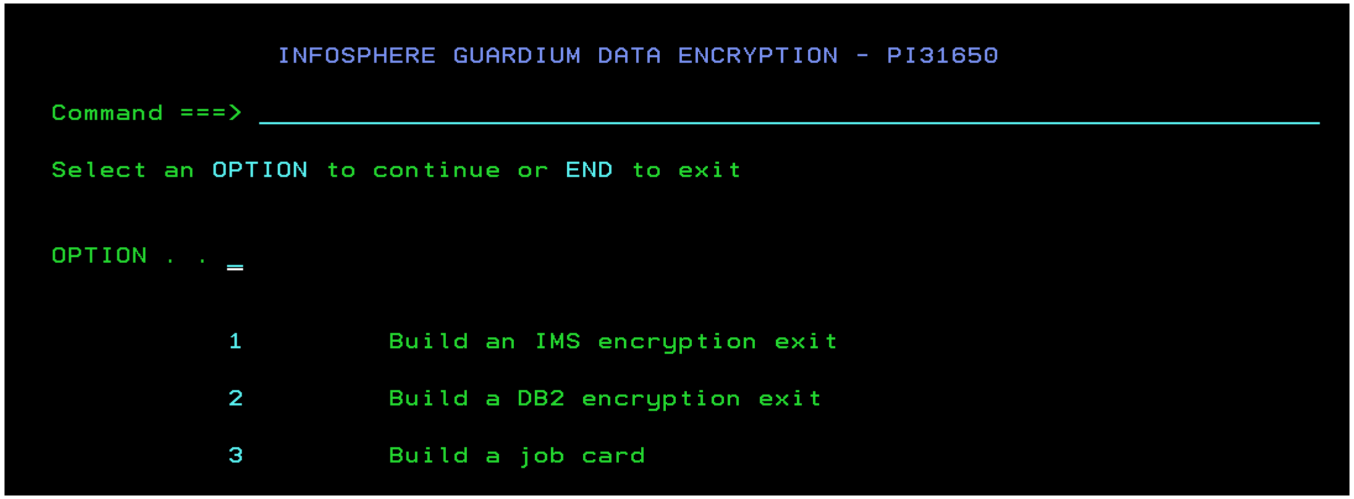


```
                 INFOSPHERE GUARDIUM DATA ENCRYPTION - PI31650

Command ===> ______________________________________________________________

Press ENTER to continue or END to exit

Jobcard    . . //DDS0027R JOB USER=&SYSUID,NOTIFY=&SYSUID,MSGCLASS=H,
           . . //        REGION=0M,TIME=5
           . . //*
           . . 
           . . 
```

Standard installation Jobcard information

Database Encryption

InfoSphere GDEz Data Encryption – ISPF Main Menu

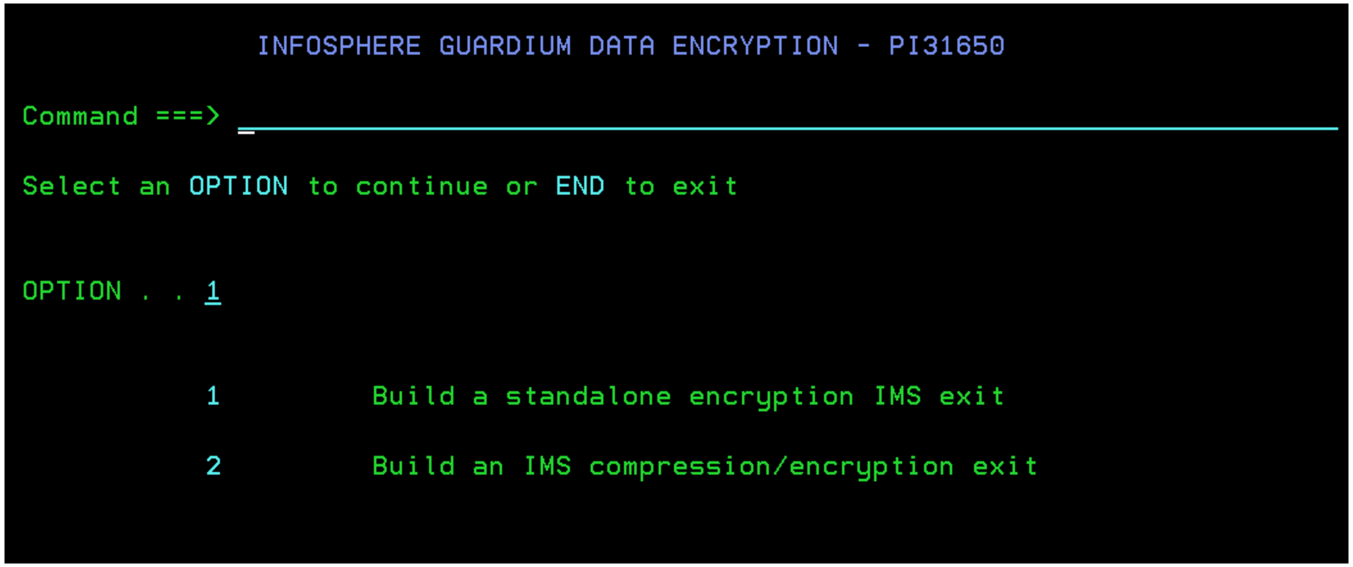


```
                 INFOSPHERE GUARDIUM DATA ENCRYPTION - PI31650

Command ===> ________________________________________________________________

Select an OPTION to continue or END to exit


OPTION . . _


           1         Build an IMS encryption exit

           2         Build a DB2 encryption exit

           3         Build a job card
```

Selection 1 for an IMS Encryption Implementation

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?

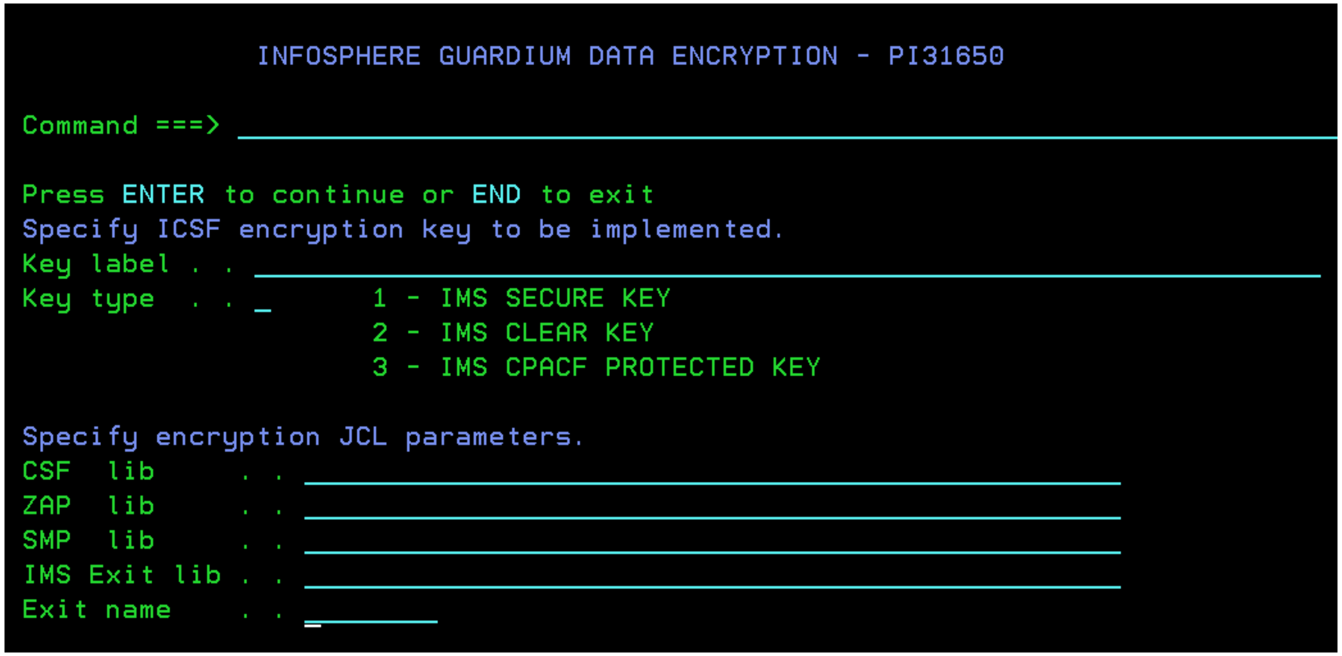


```
                INFOSPHERE GUARDIUM DATA ENCRYPTION - PI31650

Command ===> _

Select an OPTION to continue or END to exit


OPTION . . 1


           1         Build a standalone encryption IMS exit

           2         Build an IMS compression/encryption exit
```

Selections:
1 = use to create an encryption exit that will be used standalone; that is without co-existence with a compression routine
2 = use to create both an encryption exit and a driver module to call an existing compression routine then the encryption exit

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?

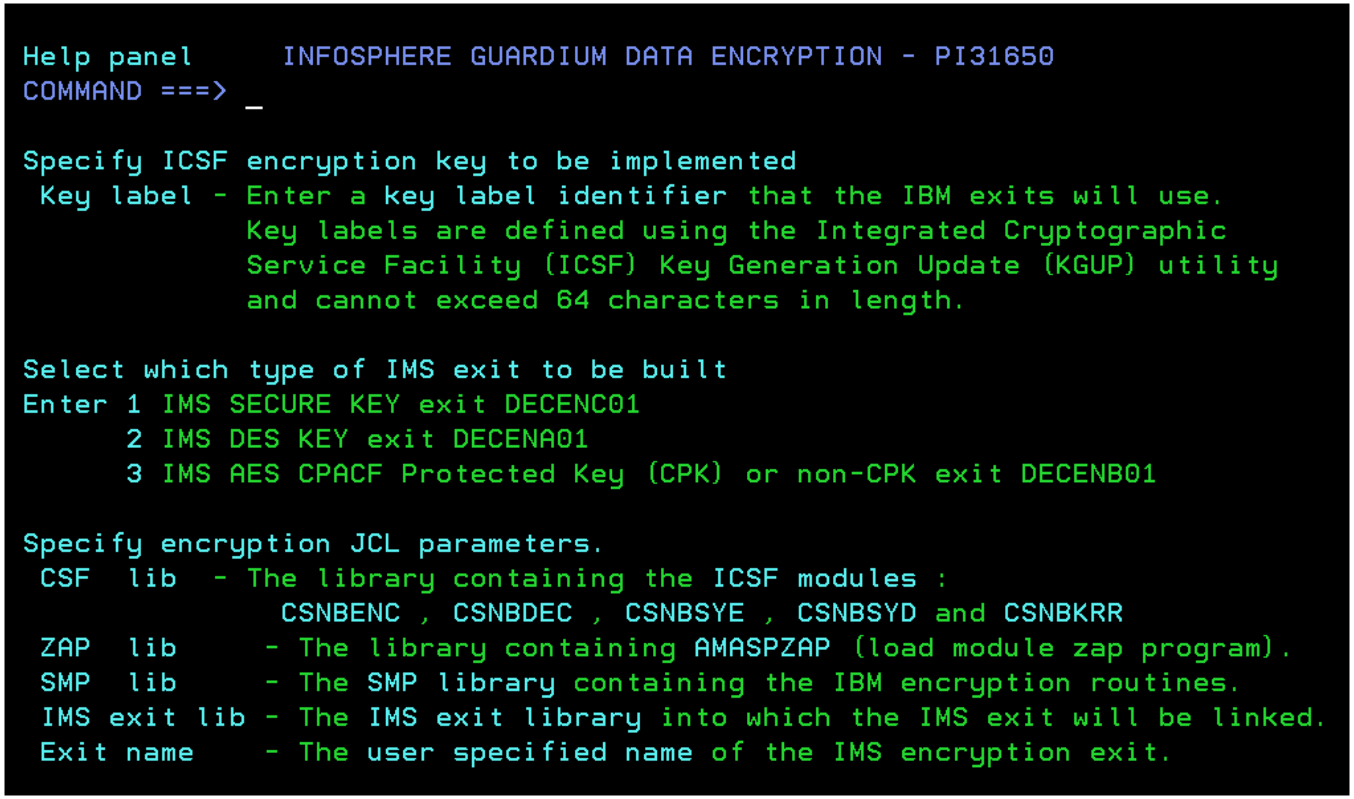


```
              INFOSPHERE GUARDIUM DATA ENCRYPTION - PI31650

Command ===> ________________________________________________________________

Press ENTER to continue or END to exit
Specify ICSF encryption key to be implemented.
Key label . . _______________________________________________________________
Key type  . . _      1 - IMS SECURE KEY
                     2 - IMS CLEAR KEY
                     3 - IMS CPACF PROTECTED KEY

Specify encryption JCL parameters.
CSF  lib      . . ________________________________________________
ZAP  lib      . . ________________________________________________
SMP  lib      . . ________________________________________________
IMS Exit lib  . . ________________________________________________
Exit name     . . ________
```

The F1 key provides help information for the screen displayed.

Database Encryption

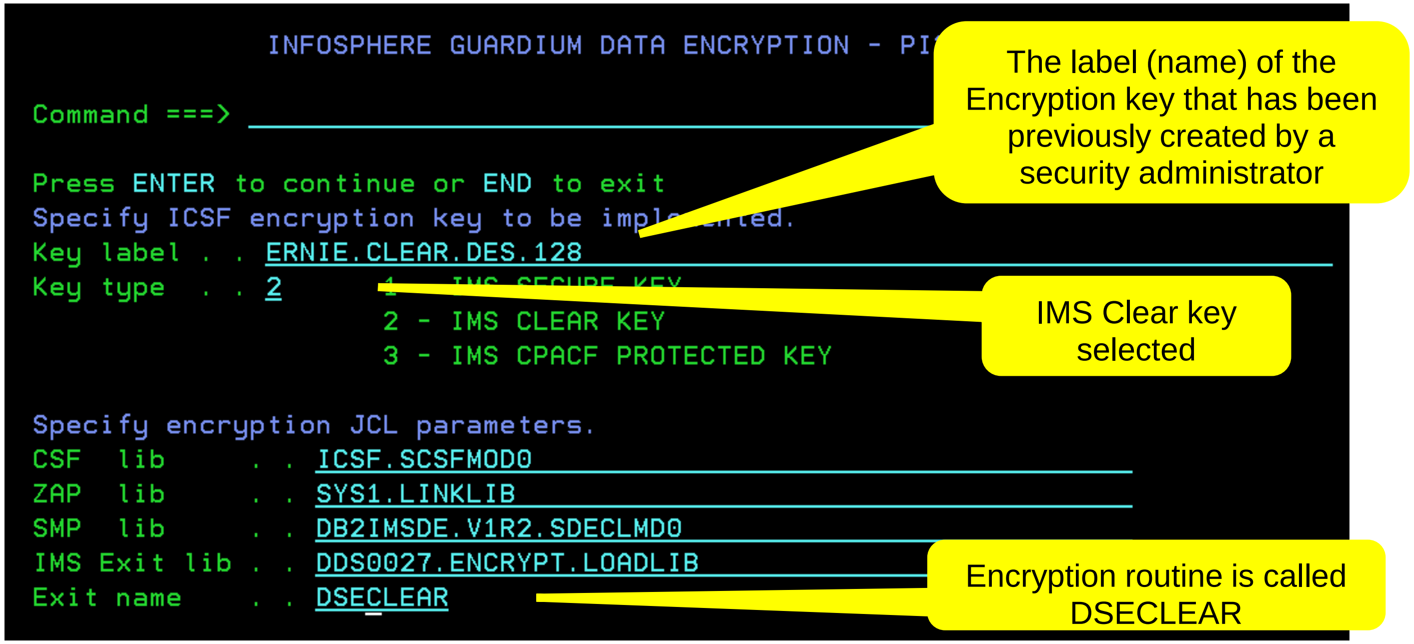
How is crypto Implemented with the Data Encryption Tool for IMS?



```
Help panel     INFOSPHERE GUARDIUM DATA ENCRYPTION - PI31650
COMMAND ===> _

Specify ICSF encryption key to be implemented
 Key label - Enter a key label identifier that the IBM exits will use.
             Key labels are defined using the Integrated Cryptographic
             Service Facility (ICSF) Key Generation Update (KGUP) utility
             and cannot exceed 64 characters in length.

Select which type of IMS exit to be built
Enter 1 IMS SECURE KEY exit DECENC01
      2 IMS DES KEY exit DECENA01
      3 IMS AES CPACF Protected Key (CPK) or non-CPK exit DECENB01

Specify encryption JCL parameters.
 CSF  lib   - The library containing the ICSF modules :
               CSNBENC , CSNBDEC , CSNBSYE , CSNBSYD and CSNBKRR
 ZAP  lib     - The library containing AMASPZAP (load module zap program).
 SMP  lib     - The SMP library containing the IBM encryption routines.
 IMS exit lib - The IMS exit library into which the IMS exit will be linked.
 Exit name    - The user specified name of the IMS encryption exit.
```

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?



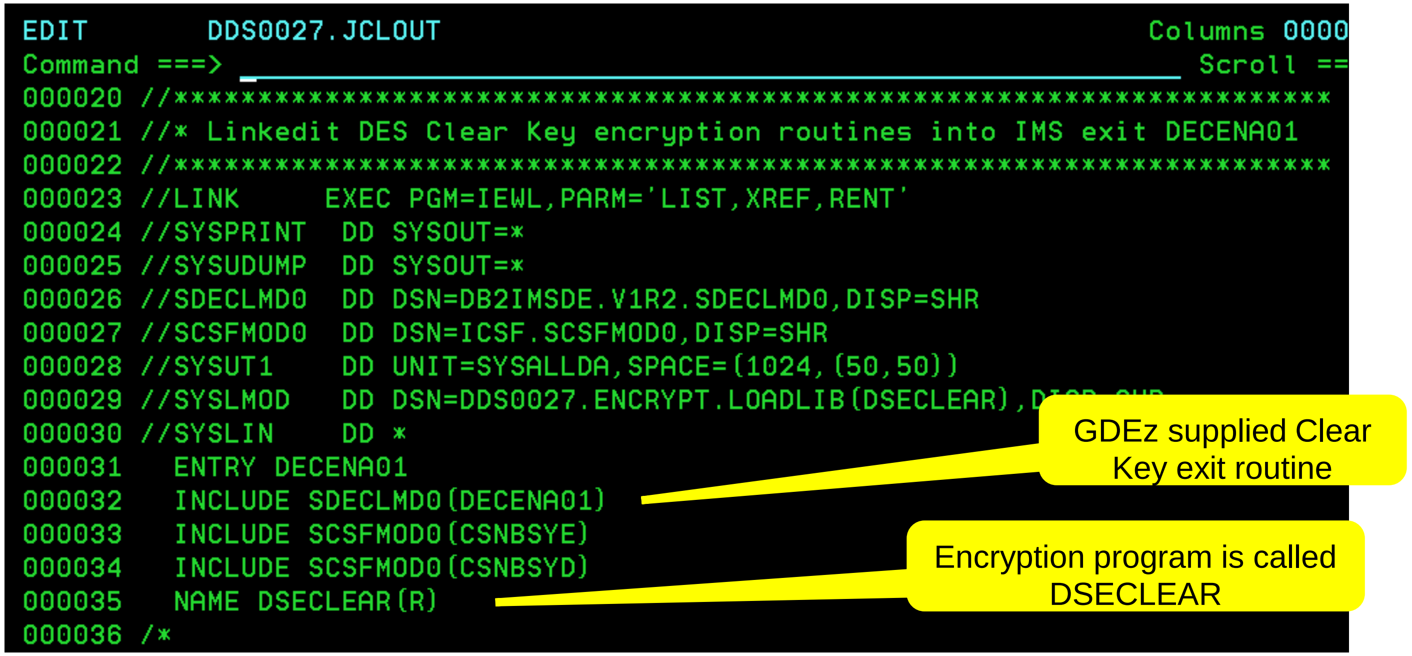
```
             INFOSPHERE GUARDIUM DATA ENCRYPTION - PI

Command ===> ____________________________________

Press ENTER to continue or END to exit
Specify ICSF encryption key to be impl    nted.
Key label  . .  ERNIE.CLEAR.DES.128
Key type   . .  2       1 - IMS SECURE KEY
                        2 - IMS CLEAR KEY
                        3 - IMS CPACF PROTECTED KEY

Specify encryption JCL parameters.
CSF  lib      . .  ICSF.SCSFMOD0
ZAP  lib      . .  SYS1.LINKLIB
SMP  lib      . .  DB2IMSDE.V1R2.SDECLMD0
IMS Exit lib  . .  DDS0027.ENCRYPT.LOADLIB
Exit name     . .  DSECLEAR
```

The label (name) of the Encryption key that has been previously created by a security administrator
IMS Clear key selected
Encryption routine is called DSECLEAR

CSF lib = Installation Encryption services dataset
ZAP lib = Dataset containing AMASPZAP program
SMP lib = Installed GDEz load dataset
EXIT lib = Load dataset for the new Encryption exit
Exit Name = Load module name for the new Encryption exit

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?



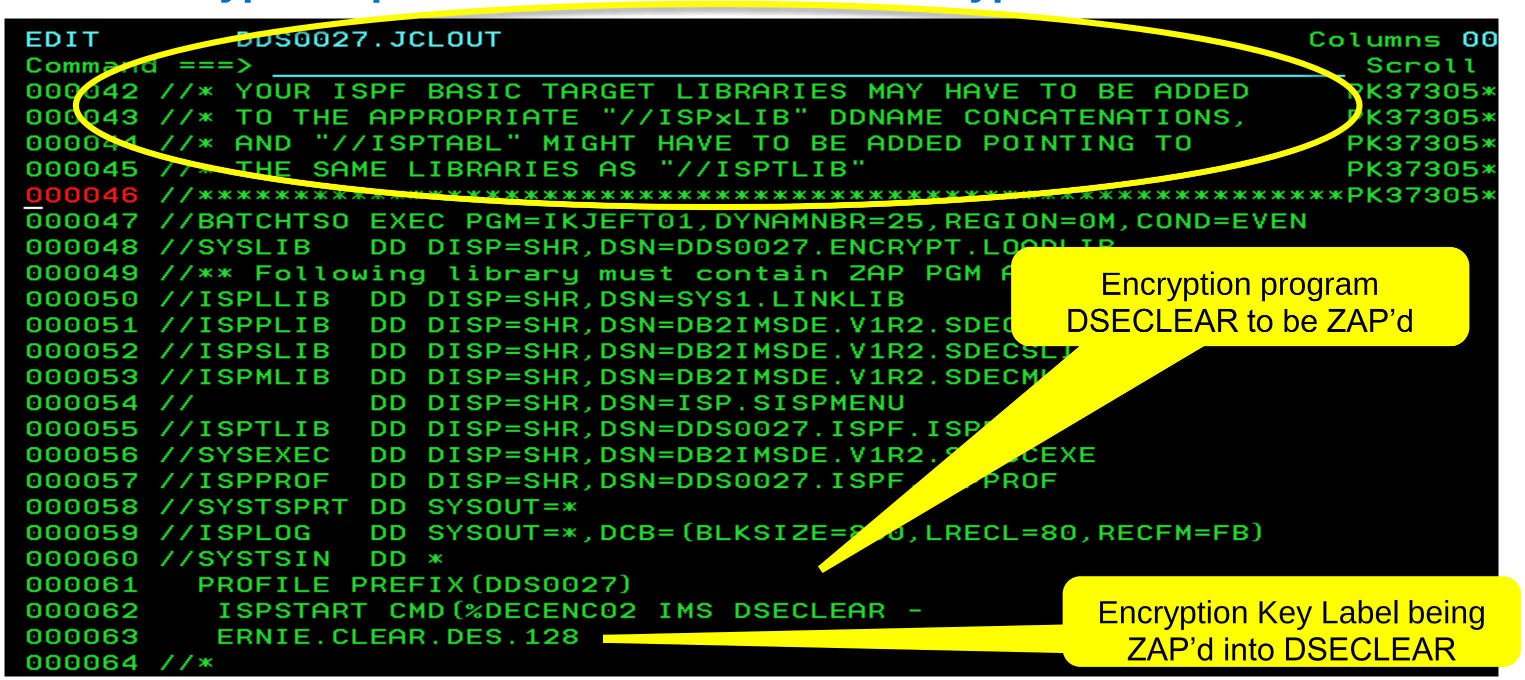
```
EDIT       DDS0027.JCLOUT                                                Columns 0000
Command ===> ________________________________________________________   Scroll ==
//******************************************************************************
//* Linkedit DES Clear Key encryption routines into IMS exit DECENA01
//******************************************************************************
//LINK     EXEC PGM=IEWL,PARM='LIST,XREF,RENT'
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SDECLMD0  DD DSN=DB2IMSDE.V1R2.SDECLMD0,DISP=SHR
//SCSFMOD0  DD DSN=ICSF.SCSFMOD0,DISP=SHR
//SYSUT1    DD UNIT=SYSALLDA,SPACE=(1024,(50,50))
//SYSLMOD   DD DSN=DDS0027.ENCRYPT.LOADLIB(DSECLEAR),D
//SYSLIN    DD *
  ENTRY DECENA01
  INCLUDE SDECLMD0(DECENA01)
  INCLUDE SCSFMOD0(CSNBSYE)
  INCLUDE SCSFMOD0(CSNBSYD)
  NAME DSECLEAR(R)
/*
```

GDEz supplied Clear Key exit routine
Encryption program is called DSECLEAR

Here is the generated JCL to create the Encryption/Decryption routine link edit. The two ICSF CSNBnnn routines are included and the resulting executable module is place into the dataset DDS0027.ENCRYPT.LOADLIB member DSECLEAR
Remember the DDS0027.ENCRYPT. LOADLIB must be in the IMS region's STEPLIB DD or the module must be copied to an existing dataset in the STEPLIB DD

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?



```
EDIT       DDS0027.JCLOUT                                          Columns 00
Command ===>                                                         Scroll
//* YOUR ISPF BASIC TARGET LIBRARIES MAY HAVE TO BE ADDED          PK37305*
//* TO THE APPROPRIATE "//ISPxLIB" DDNAME CONCATENATIONS,           PK37305*
//* AND "//ISPTABL" MIGHT HAVE TO BE ADDED POINTING TO             PK37305*
//* THE SAME LIBRARIES AS "//ISPTLIB"                              PK37305*
//*****************************************************************PK37305*
//BATCHTSO EXEC PGM=IKJEFT01,DYNAMNBR=25,REGION=0M,COND=EVEN
//SYSLIB   DD DISP=SHR,DSN=DDS0027.ENCRYPT.LOADLIB
//** Following library must contain ZAP PGM A
//ISPLLIB  DD DISP=SHR,DSN=SYS1.LINKLIB
//ISPPLIB  DD DISP=SHR,DSN=DB2IMSDE.V1R2.SDEC
//ISPSLIB  DD DISP=SHR,DSN=DB2IMSDE.V1R2.SDECSLI
//ISPMLIB  DD DISP=SHR,DSN=DB2IMSDE.V1R2.SDECML
//         DD DISP=SHR,DSN=ISP.SISPMENU
//ISPTLIB  DD DISP=SHR,DSN=DDS0027.ISPF.ISP
//SYSEXEC  DD DISP=SHR,DSN=DB2IMSDE.V1R2.S   CEXE
//ISPPROF  DD DISP=SHR,DSN=DDS0027.ISPF.   PROF
//SYSTSPRT DD SYSOUT=*
//ISPLOG   DD SYSOUT=*,DCB=(BLKSIZE=8 0,LRECL=80,RECFM=FB)
//SYSTSIN  DD *
  PROFILE PREFIX(DDS0027)
   ISPSTART CMD(%DECENC02 IMS DSECLEAR -
   ERNIE.CLEAR.DES.128
//*
```

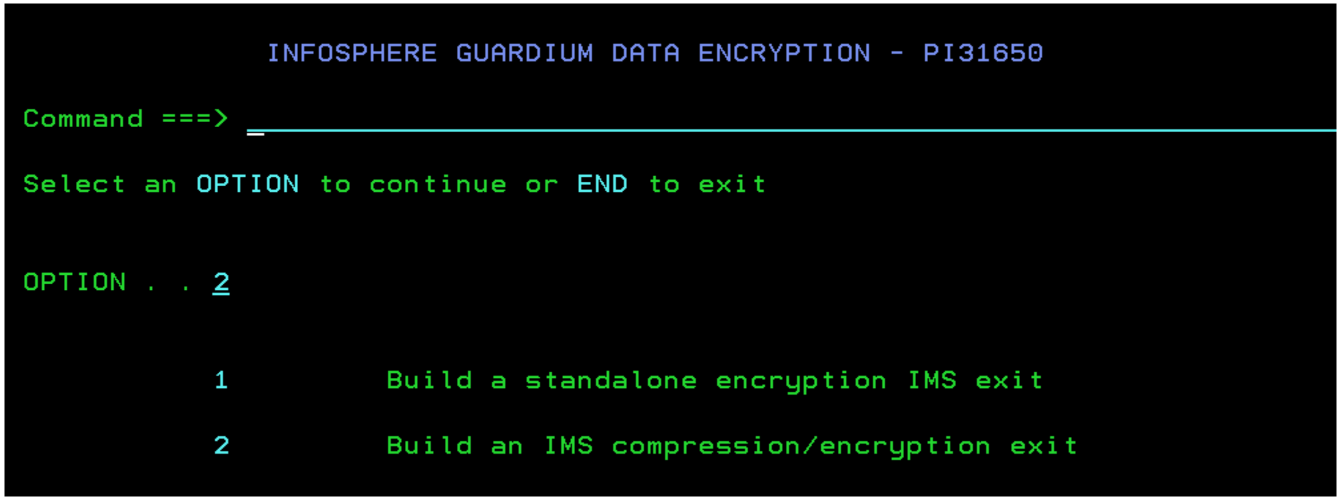
Encryption program DSECLEAR to be ZAP'd
Encryption Key Label being ZAP'd into DSECLEAR

Here is the generated JCL to create the ZAP onto Encryption/Decryption routine. Our Key Label is previously defined and resides in our ICSF dataset. This defined label is ZAP'd onto the routine providing the encryption key to be used.

The JCL may need DD concatenations added for your installation's ISPF datasets

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?



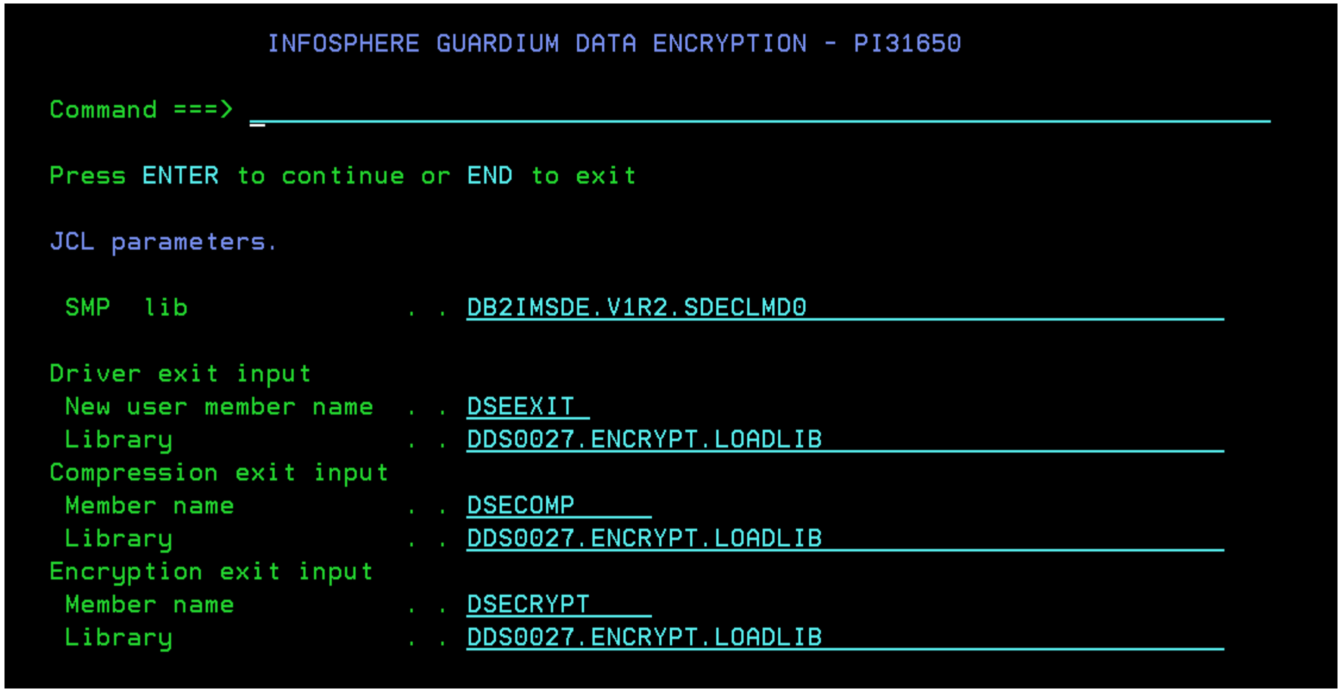
```
               INFOSPHERE GUARDIUM DATA ENCRYPTION - PI31650

Command ===> _

Select an OPTION to continue or END to exit


OPTION . . 2


           1         Build a standalone encryption IMS exit

           2         Build an IMS compression/encryption exit
```

Now let’s try a combination Compression and Encryption implementation.

The Compression and the Encryption routine must be available.

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?

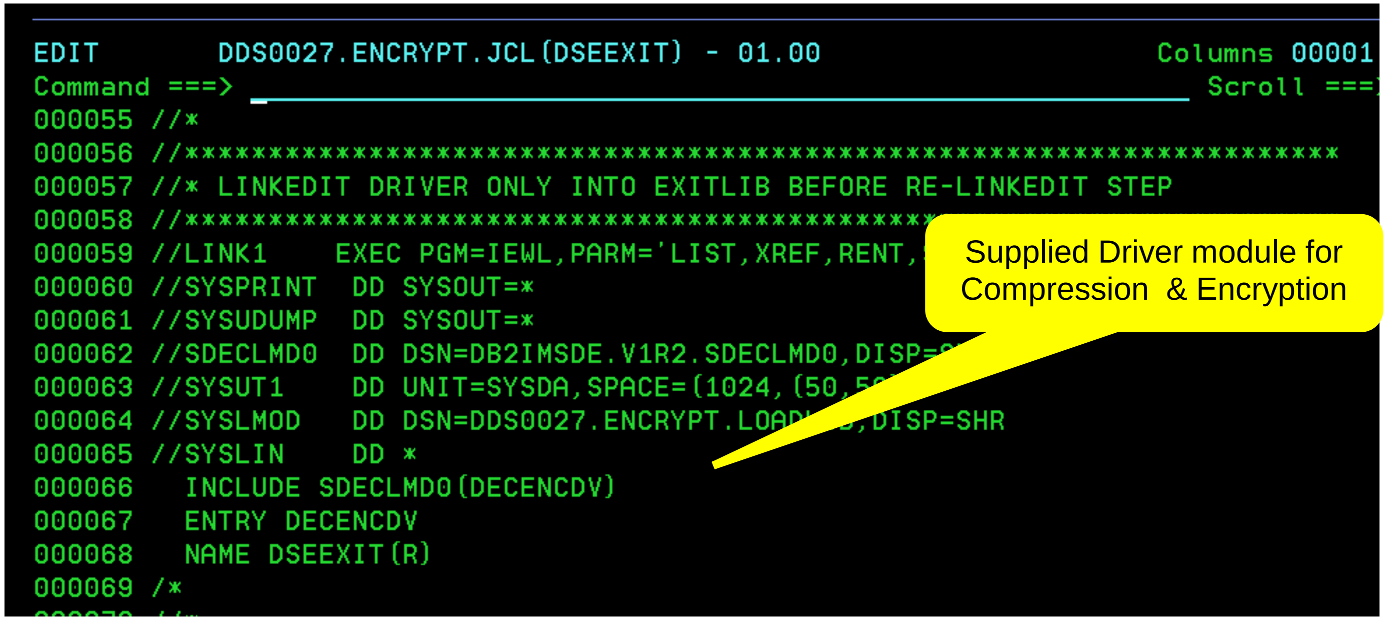


```
                INFOSPHERE GUARDIUM DATA ENCRYPTION - PI31650

Command ===> _________________________________________________________________

Press ENTER to continue or END to exit

JCL parameters.

 SMP  lib                . . DB2IMSDE.V1R2.SDECLMD0

Driver exit input
 New user member name    . . DSEEXIT
 Library                 . . DDS0027.ENCRYPT.LOADLIB
Compression exit input
 Member name             . . DSECOMP
 Library                 . . DDS0027.ENCRYPT.LOADLIB
Encryption exit input
 Member name             . . DSECRYPT
 Library                 . . DDS0027.ENCRYPT.LOADLIB
```

Here is the input to create the Link edit job for the combination module.
The Driver module will be called DSEEXIT.
It will include a Compression/Decompression routine named DSECOMP.
It will include an Encryption/Decryption routine called DSECRYPT.
Both of these modules must already exist in the named datasets.

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?

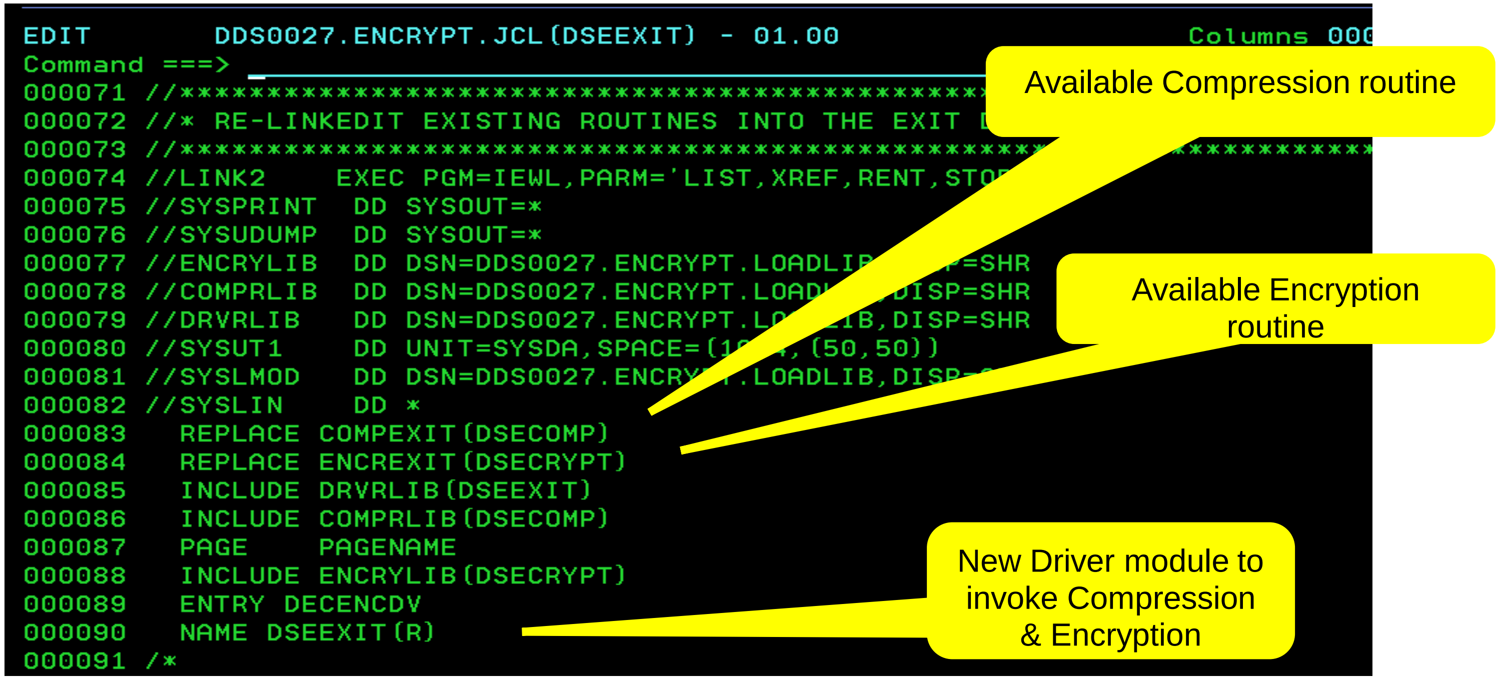


```
EDIT       DDS0027.ENCRYPT.JCL(DSEEXIT) - 01.00                    Columns 00001
Command ===>                                                            Scroll ===
//*
//*******************************************************************
//* LINKEDIT DRIVER ONLY INTO EXITLIB BEFORE RE-LINKEDIT STEP
//*******************************************************************
//LINK1    EXEC PGM=IEWL,PARM='LIST,XREF,RENT,
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SDECLMD0  DD DSN=DB2IMSDE.V1R2.SDECLMD0,DISP=S
//SYSUT1    DD UNIT=SYSDA,SPACE=(1024,(50,5
//SYSLMOD   DD DSN=DDS0027.ENCRYPT.LOAD   ,DISP=SHR
//SYSLIN    DD *
  INCLUDE SDECLMD0(DECENCDV)
  ENTRY DECENCDV
  NAME DSEEXIT(R)
/*
```

The first step of the Link edit job creates the IMS Driver module name in the target load dataset. (SYSLMOD)

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?



```
EDIT       DDS0027.ENCRYPT.JCL(DSEEXIT) - 01.00                  Columns 000
Command ===> 
//*****************************************************
//* RE-LINKEDIT EXISTING ROUTINES INTO THE EXIT
//*****************************************************            ***********
//LINK2    EXEC PGM=IEWL,PARM='LIST,XREF,RENT,STO
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//ENCRYLIB  DD DSN=DDS0027.ENCRYPT.LOADLI      P=SHR
//COMPRLIB  DD DSN=DDS0027.ENCRYPT.LOAD      DISP=SHR
//DRVRLIB   DD DSN=DDS0027.ENCRYPT.LO     IB,DISP=SHR
//SYSUT1    DD UNIT=SYSDA,SPACE=(1    ,(50,50))
//SYSLMOD   DD DSN=DDS0027.ENCRY   .LOADLIB,DISP=
//SYSLIN    DD *
  REPLACE COMPEXIT(DSECOMP)
  REPLACE ENCREXIT(DSECRYPT)
  INCLUDE DRVRLIB(DSEEXIT)
  INCLUDE COMPRLIB(DSECOMP)
  PAGE    PAGENAME
  INCLUDE ENCRYLIB(DSECRYPT)
  ENTRY DECENCDV
  NAME DSEEXIT(R)
/*
```

Available Compression routine
Available Encryption routine
New Driver module to invoke Compression & Encryption

The second step of the Link edit job includes the named Compression routine and the named Encryption routine to create the composite module named DSEEXIT in the target load dataset.

Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?

- IMS DBD update
  - COMPRTN = is added
  - The value of DATA Encrypts only the segment data
  - This value may be entered as KEY to Encrypt any segment field
  - Generate the DBD and the ACB(s)

```
                DBD
                NAME=F2O1P4,ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,10,100)
DSG001          DATASET DD1=F2O1P41,DEVICE=3380,SIZE=(8192),SCAN=1
*
                 SEGM  NAME=ROOT,BYTES=20,PTR=(TB),
                    PARENT=0,COMPRTN=(DSEEXIT,DATA,INIT)
                FIELD NAME=(ROOTKEY,SEQ,U),BYTES=10,START=1,TYPE=C
                FIELD NAME=ROOTFLD1,BYTES=1,START=4,TYPE=C
                FIELD NAME=ROOTFLD2,BYTES=1,START=5,TYPE=C
```

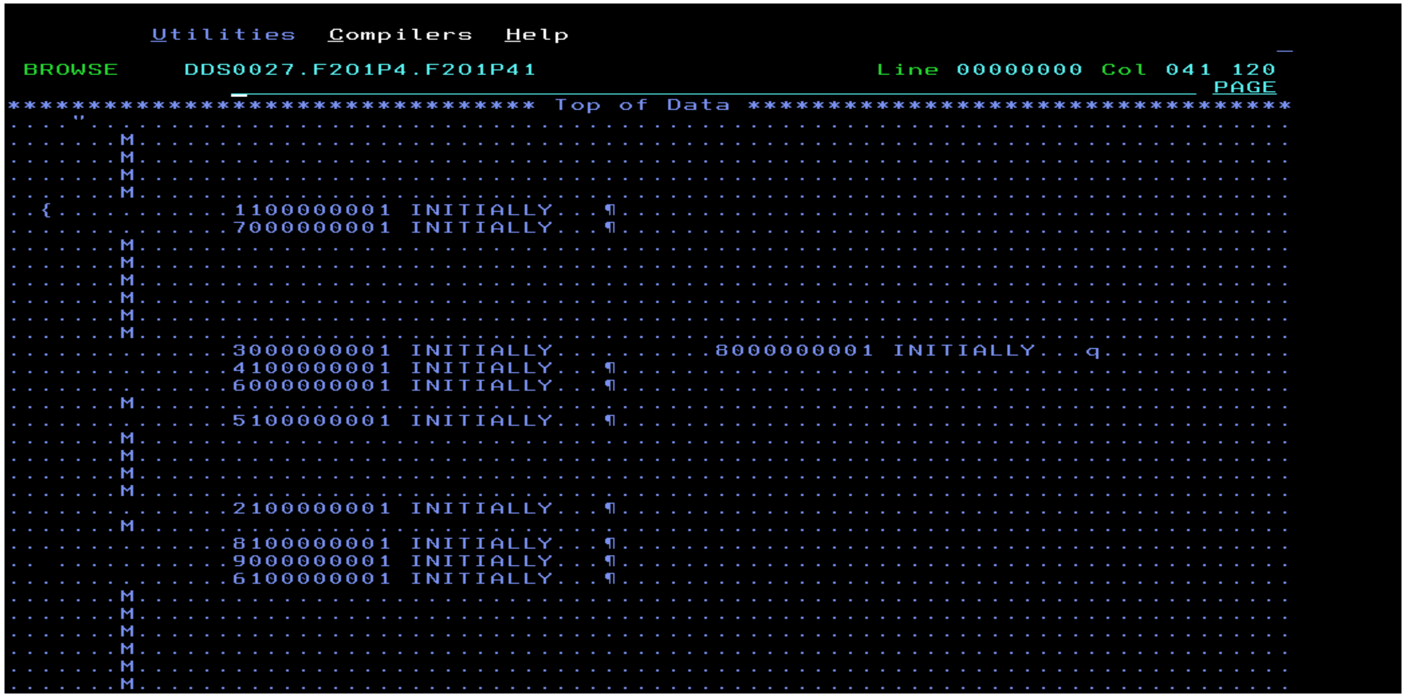
Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?

- Implementing IMS Encryption with the Data Encryption Tool
  - ⇒ Generate Key using ICSF KGUP (Key Generation Update Program)
  - ⇒ Prepare your exit using Data Encryption Tool providing ICSF Keylabel
  - ⇒ Generate the DBD and ACB(s) to include the COMPRTN value
  - ⇒ Unload target database
  - Activate the ACB to your IMS systems
  - LOAD the target database
  - /STA *database*
  - Encryption is now operational

Database Encryption

CLEAR IMS DATA



```
         Utilities  Compilers  Help
 BROWSE     DDS0027.F201P4.F201P41                      Line 00000000 Col 041 120
                                                                           PAGE
********************************* Top of Data **********************************
...."...........................................................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
..{...........1100000001 INITIALLY...¶..........................................
..............7000000001 INITIALLY...¶..........................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
..............3000000001 INITIALLY..........8000000001 INITIALLY...q............
..............4100000001 INITIALLY...¶..........................................
..............6000000001 INITIALLY...¶..........................................
.......M........................................................................
..............5100000001 INITIALLY...¶..........................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
..............2100000001 INITIALLY...¶..........................................
.......M........................................................................
..............8100000001 INITIALLY...¶..........................................
..............9000000001 INITIALLY...¶..........................................
..............6100000001 INITIALLY...¶..........................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
.......M........................................................................
```

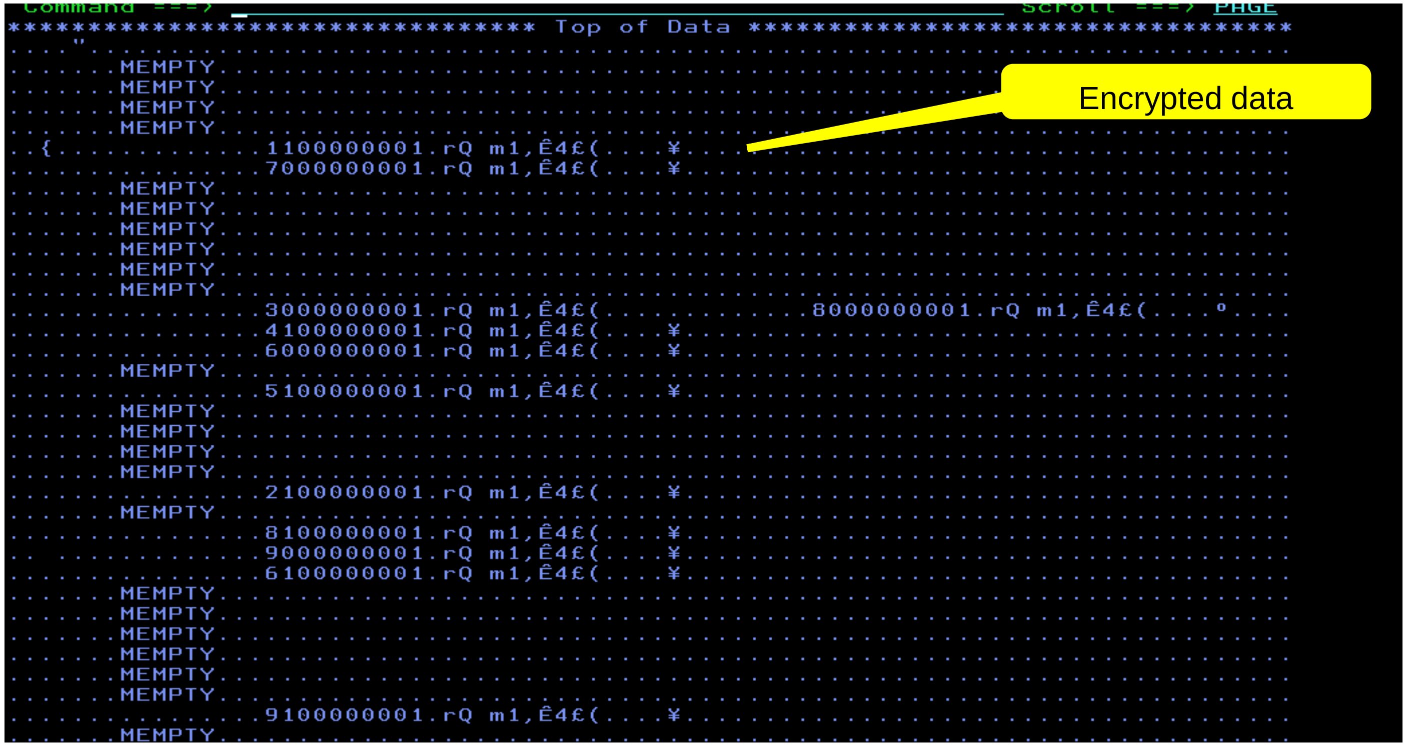
Database Encryption

How is crypto Implemented with the Data Encryption Tool for IMS?

- Implementing IMS Encryption with the Data Encryption Tool
  - Generate Key using ICSF KGUP (Key Generation Update Program)
  - Prepare your exit using Data Encryption Tool providing ICSF Keylabel
  - Generate the DBD and ACB(s) to include the COMPRTN value
  - Unload target database
  - Activate the ACB to your IMS systems
  - LOAD the target database
  - /STA *database*
  - Encryption is now operational

Database Encryption

ENCRYPTED IMS DATA



```
 Command ===>                                              Scroll ===> PAGE
********************************* Top of Data **********************************
....".........................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
..{.............1100000001.rQ m1,Ê4£(....¥......................................
................7000000001.rQ m1,Ê4£(....¥......................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
................3000000001.rQ m1,Ê4£(.............8000000001.rQ m1,Ê4£(....º....
................4100000001.rQ m1,Ê4£(....¥......................................
................6000000001.rQ m1,Ê4£(....¥......................................
.......MEMPTY...................................................................
................5100000001.rQ m1,Ê4£(....¥......................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
................2100000001.rQ m1,Ê4£(....¥......................................
.......MEMPTY...................................................................
................8100000001.rQ m1,Ê4£(....¥......................................
................9000000001.rQ m1,Ê4£(....¥......................................
................6100000001.rQ m1,Ê4£(....¥......................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
.......MEMPTY...................................................................
................9100000001.rQ m1,Ê4£(....¥......................................
.......MEMPTY...................................................................
```

Encrypted data

90

Database Encryption

Encryption Implementation Considerations

- Data Administrator - Data Encryption Tool
  - Sets up the EDITPROC and specifies the key to be used for the entire table

    Or
  - Sets up the Segment exit routine
  - Key must be defined to/managed by ICSF (stored in the CKDS)
- Application
  - Application logic determines which key to use for each field/column or segment
  - Password is managed by the application
- Security requirements
- Performance requirements
- Application/production support
- Space considerations
- Crypto hardware available

Database Encryption

Encryption Algorithms – Which Ones Are Best?

- 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-Rev1/SP-800-67-Rev1.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