Pervasive Encryption

IBM Z Platform Evaluation Test Experiences

A Showcase of zPET's Experiences with Pervasive Encryption
Within a Customer-Like Environment

Authors: Danielle Judka, Susan Marcotte, Kieron Hinds, Azeem Mohammed
With Contributions from the IBM zPET team

INTERNATIONAL BUSINESS MACHINES | IBM

Table of Contents:

1. Introduction to Pervasive Encryption
2. Introduction to zPET Environment and Why We're Using It
3. Pervasive Encryption: Setup
    a. Hardware + Software Requirements for Setup
    b. Setting up data set encryption in our environment
    c. Testing Data Set Encryption
    d. Setting Up CF Encryption

4. Managing An Environment With Pervasive Encryption
    a. Operating z/OS Systems With Pervasive Encryption
    b. Dump and Restore encrypted data between two Parallel Sysplexes Operating
    c. Accessing The Data Key On Multiple CKDS'
    d. Using XMIT to transfer data in clear format
    e. Operating Parallel Sysplex With Encrypted CF Structure Data

5. Pervasive Encryption: Middleware
    a. Middleware Experiences

6. Other Useful Tooling For Pervasive Encryption
    a. IBM zBNA for Pervasive Encryption Estimation
    b. IBM zSecure

7. Helpful Documentation

8. Summary
    a. Conclusion + Retrospective
    b. Acknowledgements

Introduction to Pervasive Encryption for IBM Z®

The IBM z14 (z14) platform introduced new encryption capabilities designed to enable protection of IBM Z data efficiently in flight and at rest as part of the Pervasive Encryption solution. A critical capability that IBM made available is z/OS Data Set Encryption on z/OS V2R2 and z/OS V2R3. This function is available through the base Data Facility Storage Management Subsystem (DFSMS™) component of z/OS. Data Set Encryption enables encryption through the DFSMS access methods. z/OS data set encryption provides the ability to encrypt the following types of data sets:

- Sequential extended format data sets, accessed through BSAM and QSAM
- VSAM extended format data sets (KSDS, ESDS, RDS, VRRDS, LDS), accessed through base VSAM and VSAM RLS

Encrypted data sets must be SMS-managed extended format. They also can be compressed format.

Along with Data Set Encryption, z/OS V2R3 introduces the ability to encrypt coupling facility (CF) structure data. With this support, all customer information stored in CF structures can be encrypted while being transferred to and from the CF and while it resides at rest in the CF. CF List and Cache structures can contain customer data and can therefore be encrypted while Lock structures as well as Directory Only Cache structures do not and will not allow encryption.

There are several reasons why it may make sense to encrypt data sets and CF structure data. These reasons include the direction with data protection standards like the General Data Protection Regulation (GDPR) that encryption and other security measures to protect all personal data are expected to be utilized or face financial and reputational consequences when a data breach occurs. Furthermore, encryption and decryption is handled transparently by the z/OS host, meaning there are no application or middleware enablement support requirements to accomplish this data encryption. Additionally, since the encryption and decryption processing is done by IBM Z Server's Protected key CPACF feature which provides high performance low latency encryption, there should be minimal impact to processing when the data is encrypted.

Introduction to zPET Environment: Why We're Using Pervasive Encryption

The zPET environment consists of running data sharing Parallel Sysplexes on the z Systems platform in as much of a customer-like fashion as possible. In this environment, we run the latest z/OS release along with customer like applications. The concept behind this testing, is that if two software products are going to run together on the same operating system platform, they have to be tested together with a focus on their interactions.

We are the final verification of a z/OS release prior to its becoming generally available to customers. We ensure that all elements and features of z/OS work seamlessly together and can support true production, mission-critical work. We also verify that z/OS provides all the industrial-strength z/OS advantages you're depending on: reliability, availability, and serviceability.

We focus on providing availability of applications to end users, and we pay attention to service level agreements and performance objectives. We look at the recovery aspects and behavior of our systems from an end user's perspective.

So, other than the obvious reason that we should test it as part of our z/OS responsibility, why did we implement Pervasive Encryption? For us, enabling any one technology pervasively doesn't always make sense especially as we try to evaluate the many deliverables for z/OS in our single environment. In the case of encryption, up until now there was the added complexity involved in configuration and management of encrypted data not to mention the system resource. Selective encryption has therefore been our approach. Now, with this new encryption capability we see many scenarios where we can use encryption pervasively. We have customer like workloads interacting with components across the Z software platform running on the latest z Systems in data sharing Parallel Sysplexes. And we like to share our experiences. In fact, this document is a compilation of our experiences that we have blogged about at our website at http://ibm.biz/zPETBlog.
So we might be a bit biased but we couldn't think of a better environment to setup and test Pervasive Encryption out in before you do!

Pervasive Encryption: Setup

Hardware and Software Requirements for Pervasive Encryption

The "**Pervasive Encryption Frequently Asked Questions**" document **ZSQ03116-USEN-02** available at: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=ZSQ03116USEN provides the most up to date information about the minimum and recommended hardware and software required for Pervasive encryption.

We provide screen captures of those requirements as supported at the time of this document's publication below for your easy reference but please always refer to the actual document for the most up to date list.

Minimum and recommended hardware and software

The minimum and recommended hardware and software for z/OS data set encryption is as follows:

Product/Feature	Required Level	Description
Hardware (HW)		
Minimum HW	z196 CPACF	Minimum HW
	Crypto Express3 or higher	Minimum HW for Secure-key/Protected-key CPACF[1]
Recommended HW	z14 CPACF	CPACF performance improvements
	z14	Enterprise Key Management Foundation (EKMF) feature
Operating System Support		
DFSMS	z/OS 2.3	Full support in 2.3, 2.2 plus service
	z/OS 2.2 + OA50569 PTFs	
	z/OS 2.1 + OA50569 PTFs	Toleration only – read/write, cannot create encrypted data sets.
RACF®	z/OS 2.3	DFP segment key label; conditional access checking
	z/OS 2.1, 2.2 + OA50512 PTFs	
ICSF	HCR77C0 or HCR77C1 when available	Protected-Key Read
	HCR77A0–B1 + OA50450 PTFs	
1– Secure-key is *strongly recommended* for production environments. Clear key can be used in dev/test.		

The minimum and recommended hardware and software for coupling facility data encryption is as follows:

Product/Feature	Required Level	Description
Hardware		
z/OS: Minimum HW	zEC12	Minimum supported for z/OS 2.3
	Crypto Express3	Required for Protected-key CPACF
z/OS: Recommended HW	z14	AES-CBC CPACF encrypt/decrypt performance improvements
CF: Recommended HW	z14	Simplified recovery for specific sysplex-wide reconciliation scenarios
Operating System – Base Support		
z/OS	z/OS 2.3	z/OS support for CF encryption
Additional Support		
zSecure®	zSecure 2.3	zSecure Audit support for CF encryption
zBNA	zBNA	zBatch Network Analyzer support for CF encryption

Toleration Support

z/OS V2.1 with Toleration APAR OA50569 applied is designed to support z/OS data set encryption in toleration mode, meaning it can read from and write to encrypted data sets but not create new encrypted data sets.

CF encryption is available with z/OS v2.3, and is not available for down level systems. Toleration APAR OA52060 for z/OS V2.1 and z/OS V2.2 prevents customers from using encrypted CF structures but will allow z/OS V2.2 and 2.1 systems to coexist in a sysplex with encrypted structures.

Setting Up Encryption Keys For Use During Data Set Encryption And zFS Encryption

There is minimal setup required to take advantage of data set encryption. Part of the setup is to create a key which must be used when encrypting the data. Using ICSF, a secure AES-256-bit encryption DATA key must be created and stored in the ICSF key repository (CKDS). This key is a secure key and can be referred to by the key label associated with it.

In our environment, we already have ICSF setup with access to Crypto Express cards and an AES Master Key. For those of you setting up ICSF for the first time, the IBM Crypto Education Wiki has a very informative entry which contains step by step instructions starting with configuring the crypto cards all the way to encrypting data. It also includes a very informative pitch on Key Management:

Pervasive Encryption – zOS Data Set Encryption
(https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W7df80301055d_495b_bb88_a0a2f84757c5/page/Pervasive%20Encryption%20-%20zOS%20Data%20Set%20Encryption)

During the testing with z/OS V2R2 and the initial testing with z/OS V2R3, we were running ICSF HCR77C0. We used a REXX exec to create the AES-256 bit DATA key. The exec used was very similar to the exec contained in *Pervasive Encryption – zOS Data Set Encryption (Step 6 Generate a Secure AES Data key).* This secure DATA key is encrypted under the AES master key that is set in the cryptographic coprocessor.

Once we migrated to ICSF HCR77C1 we used a new function to create the secure AES-256 bit DATA keys. Through the ICSF panels you can now easily create a secure AES-256 bit DATA key. From the ICSF main panel, *Integrated Cryptographic Service Facility*, we selected option 5 UTILITY. From the *ICSF – Utilities* panel, we selected option 5 *CKDS KEYS*. On the *ICSF – CKDS KEYS* panel, we selected option 7 *Generate AES DATA keys*. On the *ICSF – CKDS Generate Key* panel, we then entered a key label and selected 256 for the AES key bit length. Once enter is pressed, the AES DATA key is created and can be accessed using the key label specified.

After creating the secure DATA key, we then setup protection for the key in the CKDS and also configured it to be used as a protected key. Using RACF, we created a profile using the following command where DATA KEY LABEL is the key label you used to create the key above:

RDEF CSFKEYS *DATA KEY LABEL* UACC(NONE) ICSF(SYMCPACFWRAP(YES) SYMCPACFRET(YES))

We then gave the appropriate user ID, in this case, *OWNERID*, access to the key via the key label *DATA KEY LABEL*:

PERMIT *DATA KEY LABEL* CLASS(CSFKEYS) ID(*OWNERID*) ACCESS(READ)
WHEN(CRITERIA(SMS(DSENCRYPTION)))

By specifying, **WHEN(CRITERIA(SMS(DSENCRYPTION)))**, we are indicating the ***OWNERID*** can only use this key when performing data set encryption.

In order for the changes to CSFKEYS to take affect a **SETROPTS RACLIST(CSFKEYS) REFRESH** needs to be issued.

Note that the *Pervasive Encryption – zOS Data Set Encryption Step 7 Protect Data Sets with Secure Keys* entry in the IBM Crypto Education Wiki, contains an exec which issues commands similar to what I show above.

In our environment, we run with ICSF option **CHECKAUTH(YES)** which indicates to ICSF to issue **RACROUTE** calls to perform security access control checking. In other words, ICSF will be checking to see if the userid attempting to use the key or the ICSF service has the appropriate access. If you run with **CHECKAUTH(NO)**, no security access control checking through the **RACROUTE** call will take place. Note that if you are running with ICSF HCR77C0 or above, you can change this options value dynamically using the **SETICSF** command.

We run with multiple CKDS' in our ICSF environment. Since our workloads run across various images in our plex, we wanted to ensure that the data key was available on all images. Because of this, we placed the encrypted data key in all CKDS'. We were able to do this as we run w/ the same AES Master across all cards in our plex. On the image we originally created the encrypted data key on, we used the ICSF callable service, CSNBKRR, and the key label to obtain the encrypted data key.

We then used this encrypted data key output on the image using the other CKDS to add the key to the CKDS. Specifically, we used the ICSF callable service, CSNBKRC2, plus the key label and the encrypted data key as input. This service will place the encrypted data key in the CKDS under the key label specified in the CSNBKRC2 call.

Note: CF encryption does not require the manual setup of a key. The administrative data utility will create and assign secure cryptographic key tokens to a structure definition in the CFRM Couple Data set (CDS).

We tested encryption with the following methods:

- Using DFSMS ACS routines that specify the data key
- Using DFSMS ACS routines that do not specify the data key
- Using the key label parameter in the JCL

Using DFSMS ACS routines that specify the data key

1) We created RACF FACILITY profile **STGADMIN.SMS.ALLOW.DATASET.ENCRYPT** and gave READ access to the users who would be encrypting data sets. This allows the SMS ACS routines that specify a key label to be used when creating encrypted data sets.

2) We created separate DFSMS Data Classes for each component (Db2,IMS,CICS etc.) and performed the following
   a. Assigned a unique data key label to each Data class.
   b. The "Data Set Name Type" in Data Class also needs to be in "EXT" format. In other words, encryption only works when data sets (PS or VSAM) are in extended format.

3) We updated our SMS Data Class routines and assigned VSAM or PS data sets to the appropriate data class and ensured that data sets are encrypted upon allocation with the IDCAMS LISTCAT command.

Using DFSMS ACS routines that do not specify the data key

1) We added DFP segments to the RACF data set profiles protecting the data sets we wanted to encrypt. The DFP segments specify the key label of the data key to use. We used the following command to add the key label to the DFP segment:
   ALD <data set profile> DFP(DATAKEY(<key label>) RESOWNER(<data set owner>))

2) We created separate DFSMS Data Classes for each component (DB2,IMS,CICS etc.). Data set encryption only works when data sets (PS or VSAM) are in extended format, therefore the "Data Set Name Type" in the Data Class needs to be in "EXT" format.

3) We updated our SMS Data Class routines and assigned VSAM or PS data sets to appropriate data class and ensured that data sets are encrypted upon allocation with IDCAMS LISTCAT command

Using the key label parameter in the JCL

To allocate encrypted data sets without modifying ACS routines, we added **DSKEYLBL=”key label”** and **DSNTYPE=EXTREQ** in the JCL along with other allocation attributes and tested encryption on sequential files successfully with the IDCAMS LISTCAT command.

```
//ALLOPSFL JOB ,
//    MSGLEVEL=(1,1),MSGCLASS=H,CLASS=A,NOTIFY=&SYSUID
//*****************************************************************
//* THIS IS A SUCCESSFUL ALLOCATION OF FLAT FILES.                **
//*****************************************************************
//STEP010     EXEC PGM=IEFBR14,TIME=5
//OUTPUT1  DD DSN=DATASET NAME,
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=8000),
//         SPACE=(TRK,(10,5),RLSE),UNIT=3390,
//         DISP=(,CATLG,DELETE),DSNTYPE=EXTPREF,
//         DSKEYLBL=KEY LABEL
```

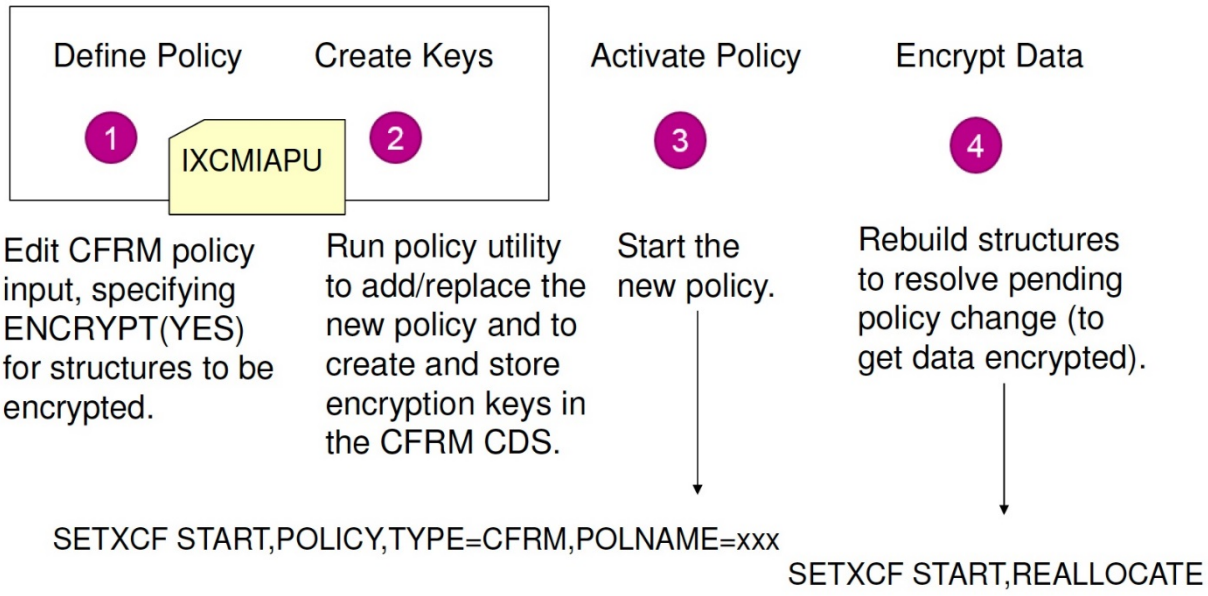
Coupling Facility exploitation in a Parallel Sysplex is a significant part of how we test z/OS and the Z system in our zPET environment and so we made CF structure encryption a central part of our test efforts for V2R3. We'd like to tell you about our approach.

Enablement approach

Encryption is enabled on a structure by structure basis via the new **ENCRYPT** structure keyword in the CFRM policy definition. The Administrative Data Utility (IXCMIAPU) is used to create or update structure definitions in the CFRM policy data, and when run with a policy that contains structures that are defined with ENCRYPT(YES), the administrative data utility will create and assign secure cryptographic key tokens to a structure definition in the CFRM Couple Data set (CDS).

These secure cryptographic key tokens are obtained from ICSF. Rebuilding a structure with a pending policy change for the ENCRYPT state is all that is required to begin encrypting or decrypting the data in that structure. Encryption of the structure data is done using the CPACF feature with the secure cryptographic key tokens associated with the structure.

Here is an infographic that summarizes the steps to enable CF structure data encryption:



Anyone familiar with how to update CFRM policy structure definitions and make those changes active will recognize that encrypting a structure takes the same standard CFRM policy update procedure. The only difference is the new **ENCRYPT** parameter, provided the requirements are in place to support it.

There are further considerations to take into account when running with CFRM policies enabled for CF structure data encryption such as managing multiple administrative policies, for example for backups and DR, or for migrating between structure or CF modes.

You must ensure that all systems associated with the sysplex, including those used to format Couple Data sets, or GDPS K-systems, or contingency LPARs, or Disaster Recovery failovers systems, all use the same AES master key for encryption. You must also run the policy utility on a system with the same AES master key as the systems in the sysplex where the encrypted structures are to be used.

Consult the z/OS 2.3.0 MVS Setting Up the Sysplex publication on exactly what needs to be considered for your specific environment and CF configuration.
Since it's required that all systems be at V2R3 before enabling CF structure encryption, we waited to enable *it* until we knew ***all*** our systems were at V2R3 permanently. We run a smaller test environment which allowed us to more easily migrate fully to V2R3, as well as test out CF structure encryption from an operational perspective, before focusing our efforts on our larger 15-way sysplex where more complex scenarios could be exercised such as data sharing applications exploiting encrypted data and heavy workload levels to stress the entire application stack while running those applications with encrypted data.

We also utilized the recently released version 1.8.1 of the IBM zBNA capacity planning tool to estimate the cost of enabling CF structure data encryption on each of our systems. Please reference the IBM zBNA section within the *Other Useful Tooling For Pervasive Encryption* for information on this tool.

Finally, we adopted a staged approach to encrypting structure data, focusing first on individual structures by type and by exploiters then broadening our focus to encrypt structures for an entire data sharing group and then finally focusing on encrypting all structures for applications.
We'll discuss the middleware structure encryption individually in the Middleware section of this document, but in summary we encrypted structures for:

- IBM IMS V14
- IBM Db2 at V11 & V12
- IBM MQ
- IBM CICS
- In addition to z/OS infrastructure support structures such as the XCF signaling, Operlog and JES2 checkpoint structures.

Operating z/OS Systems With Data Set Encryption

From here on out, we will refer to the date key encrypted under the AES Master Key as the data key.

Data Set Encryption is designed to honor specific roles in the IBM Z data center to provide data protection with separation of roles. zPET assumed the following roles as we implemented and tested Data Set Encryption.

We separated our tasks and responsibilities for managing our environment accordingly:

- **The ICSF Administrator** – Manages ICSF, responsible for key management (defining keys, key labels, changing keys etc) and working with the key management system.
- **The Security Administrator** – Responsible for creating RACF profiles, assigning access to key labels and providing encryption capabilities via RACF DS profile
- **The Storage Administrator** – Provides encryption capabilities via storage management policies (updating data classes, updating ACS routines, etc.). Manages backup, migration and replication of encrypted data sets
- **Data Owner** – Exploits encryption capabilities via JCL. Runs applications and submits jobs exploiting encrypted data sets.

Since these roles exist today, we only highlight new or improved features in z/OS that aid them in managing the z/OS environment with Data Set encryption.

The ICSF Administrator:

ICSF HCR77C1, available as a web deliverable in the fall of 2017, introduced a CKDS browser function which we were very happy to see. This CKDS browser function is available through the ICSF panel driven interface. From the main ICSF ISPF panel, Integrated Cryptographic Service Facility, choose option 5 UTILITY. This will bring you to the ICSF – Utilities panel. From this panel, option 5 CKDS KEYS is the CKDS browser function.

On this panel you can perform a variety of functions. One of the more useful ones for data set encryption is the ability to display all keys or a designated key using option 1 List and manage all records. If you want to display all keys records, then you specify option 1. If you want a specific key, you specify the key label in the Full or partial record label input field in addition to option 1. This field also allows wild cards, *, so if you're following a certain naming convention for data set encryption keys, you have the option of using up to 7 wild cards to display all data set key records.
The other very useful function on this panel is Option 7 Generate AES DATA keys. This option will allow you to create an AES 256 bit key that can be used in data set encryption. Once created, you can verify its existence and values by listing it with option 1.

The Security Administrator:

We found that the security administrator was primarily involved in setting up the data keys and key labels. Any of the RACF steps covered in the Setting up section above should be vetted and approved by the security administrator. The security administrator should also ensure that any requests for access to data keys or key labels should be approved by the data set owner.

Additionally, the security administrator should put procedures in place to monitor for access attempts to any of the facilities defined as part of setting up data set encryption, as well as access attempts to the data itself. These procedures should be well understood already by the security administrator, and should just need updating for the new facilities that data set encryption introduces. One common method which we implemented in our environment for monitoring for any access attempts and now includes data set encryption specific facilities is to automate on the ICH408I Authorization Failure messages to take action such as notifying the security administrator of such access attempts.

IBM zSecure introduces new features at V2R3 in support of Data Set encryption that makes setting up, monitoring and auditing as a security administrator easier. See the section about zSecure later in this document for more details on these new features.

The Storage Administrator:

The storage administrator role requires access to volumes and datasets for the purposes of managing them in the z/OS enterprise. We felt that the tools and procedures to do this did not change significantly, and instead needed to be adjusted to work with encrypted data sets.

For example, a commonly used command for storage management is the **IDCAMS LISTCAT** command, which is now able to show in its output whether a data set is encrypted and if so what its key label is. Here is a snippet from an **IDCAMS LISTCAT** output for an encrypted dataset showing the new fields:

```
ENCRYPTIONDATA
  DATA SET ENCRYPTION----(YES)
  DATA SET KEY LABEL-----USER.DATA.KEY
```

The Interactive Storage Management Facility (ISMF) can also be used by the Storage administrator for managing encrypted data, since it has introduced a new option to show which datasets are encrypted in the **Data Set Listing panel** (ISMF Option 1) in a new **ENCRYPTION INDICATOR** column:

```
                            ENCRYPTION
       DATA SET NAME        INDICATOR
------------(2)------------ ---(43)---
USER.DSNAME.TEST1           YES
```

One important note to consider when dataset encryption is enabled is that the storage administrator should not require access to the data set content to perform his management role. They should not be given access to the data key or key label unless approved by the data owner. This is true even for storage administrators with IDs that have the **OPERATIONS** access which allows privileged actions, including browsing non-encrypted data sets. With data set encryption, an **OPERATIONS ID** will not be able to browse an encrypted dataset unless granted access to the data key that was used to encrypt the dataset.

The Data Owner:

We found that the data owner had seamless interactions with the data when Data Set encryption is active. This is due to the actions of the other administrator roles. The data owner cares that the data they create is protected with Data Set Encryption, and is not readable by anyone one else other than those who the owner has authorized. By working with the other administrators, the data owner will be able to ensure protection of their data.

Dump & Restore Encrypted Data Between Two Parallel Sysplexes

We performed a special test where we dumped encrypted data in one of our sysplexes and restored it in a different sysplex. We used the DFSMSdss DUMP command to dump encrypted data sets (sequential format) in our production sysplex (PLEX1) and used DFSMSdss RESTORE to restore them in our test sysplex (PLEX2).

We placed the backup copy of the data on a volume which was accessible from both PLEX1 and PLEX2. However, the content of the backup copy was unreadable since the data sets we dumped were in encrypted format. We ensured that the ID on PLEX2 had proper RACF/ICSF access to the data set and keylabel so it could perform the restore and access the data set upon restore. We also had to copy the encrypted data keys from PLEX1 to PLEX2 before restoring the data sets.

Please see the **Accessing the Data Key on multiple CKDS'** section below for the procedure we used to copy the data keys from one PLEX to another.

We used the following job to dump the encrypted data sets.

```
//DUMP     EXEC PGM=ADRDSSU,REGION=4M
//SYSPRINT DD  SYSOUT=*
//DISKOUT  DD UNIT=3390,DSN=AZEEM.DSS.DUMP.SAMENC,
//         DISP=(NEW,CATLG),SPACE=(CYL,(5,1)),VOL=SER=SHR001
//SYSIN     DD  *
   DUMP OUTDDNAME(DISKOUT) TOL(ENQF)  OPT(4) COMPRESS -
   DATASET(INCL(AZEEM.ENCSAM.**))
```

The following job was used to restore the encrypted data sets.

```
//DASDIN     DD  UNIT=3390,DISP=SHR,
//         DSN=AZEEM.DSS.DUMP.SAMENC
//DASDOUT    DD  UNIT=SYSALLDA,VOL=SER=P2SMS1,DISP=SHR
//SYSIN      DD  *
    RESTORE INDD(DASDIN) OUTDD(DASDOUT) TOL(ENQF)      -
     DATASET(INCL(**)) STORCLAS(STANDARD)
/*
```

Accessing The Data Key On Multiple CKDS'

The data key used in Pervasive Encryption of Data Sets is encrypted under the AES Master Key and stored in the ICSF CKDS data set. Among our sysplexes and even images, we run with multiple CKDS'. So, when we originally encrypted the data sets on PLEX1, we created a data key to be used in the encryption. In order for the data set to be accessed successfully on PLEX2 for the RESTORE, we had to ensure that the data key was available on that image. For the data key to be available to the image on PLEX2, we had to add it to the CKDS being used by that PLEX2 image.

We were able to do this using the ICSF services below because we run with the same AES Master across all cards in both our plexes. On the PLEX1 image we originally created the data key on, we used the ICSF callable service, CSNBKRR, and the key label to obtain the encrypted data key. On the PLEX2 image we planned to run the RESTORE on, we used the ICSF callable service, CSNBKRC2, to place the data key in the CKDS. We ran this ICSF service using the encrypted data key output and the key label. This service, CSNBKRC2, places the encrypted data key in the CKDS under the key label specified in the call.

Using XMIT to transfer encrypted data in clear format

We also learned that the XMIT statement allows you to transmit encrypted records from one JES2 node to either another JES2 node or an eligible non-JES2 node in clear format.

In other words, if you want to send an encrypted file from one sysplex to another sysplex you don't need to de-encrypt it first. You can just XMIT the file and it will decrypt it under the cover for you and the receiver will get the file in clear format.

Note to Reader: On the receiving system, the user needs to make sure that they have correct authority to access the target data set via the applicable RACF data set profile so that they can store the received copy of the file.

We found that having CF structures with encrypted data to be transparent in normal operations in our parallel sysplexes. Structure policy changes, including changing the encryption state (with the new **ENCRYPT(YES | NO | default=NO))** policy keyword works seamlessly via the policy utility, then running the SETXCF START,POLICY command then rebuilding the affected structure.

We had no issues with managing and switching multiple policies that contain structures with encrypted data. We also saw no difference in switching in or out Couple Data Sets that contain those policies. We also were able to transparently change secure key tokens for structures with the **SETXCF MODIFY,STRNM=strname,ENCRYPTKEY** command.
We found that we could monitor the state of encrypted structures in several ways including:

- **DISPLAY XCF,STRUCTURE** command
- RMF Coupling Facility Structure Details dialog in RMF Monitor III
- RMF CF Activity post processor report

Here's a sample output from the **D XCF,STR** command for one of our XCF signaling structure which has been enabled for encryption. Note from the highlighted line that the POLICY information shows **ENCRYPT : YES** and the allocation information in the ACTIVE STRUCTURE reports an **ENCRYPTION LEVEL: AES-256 PROTECTED KEY** indicating that the policy definition requests encryption for this structure and that the structure has been rebuilt to enable encryption of the data sent to and stored in it.

```
D XCF,STR,STRNM=IXCPLEX_PATH6
IXC360I  15.09.09  DISPLAY XCF 978
STRNAME: IXCPLEX_PATH6
 STATUS: ALLOCATED
 EVENT MANAGEMENT: POLICY-BASED
 TYPE: LIST
 POLICY INFORMATION:
  POLICY SIZE    : 180224 K
  POLICY INITSIZE: 95232 K
  POLICY MINSIZE : 71424 K
  FULLTHRESHOLD  : 90
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: N/A
  DUPLEX         : DISABLED
  ALLOWREALLOCATE: YES
  PREFERENCE LIST: CF4      CF3      CF1      CF2
  ENFORCEORDER   : YES
  EXCLUSION LIST IS EMPTY
  ENCRYPT        : YES
```

```
ACTIVE STRUCTURE
----------------
 ALLOCATION TIME: 09/12/2017 17:02:41
 CFNAME         : CF4
 COUPLING FACILITY: 002964.IBM.02.0000000840F7
                    PARTITION: 14   CPCID: 00
 STORAGE CONFIGURATION     ALLOCATED          MAXIMUM      %
  ACTUAL SIZE:                    94 M            176 M   53

  SPACE USAGE     IN-USE       TOTAL    %
   ENTRIES:            2       20043    0
   ELEMENTS:          33       20003    0

 PHYSICAL VERSION: D32157DF 7FE67E92
 LOGICAL  VERSION: D32157DF 7FE67E92
 SYSTEM-MANAGED PROCESS LEVEL: NOT APPLICABLE
 DISPOSITION    : DELETE
 ACCESS TIME    : 0
 MAX CONNECTIONS: 32
 # CONNECTIONS  : 15
 ENCRYPTION LEVEL: AES-256 PROTECTED KEY

CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME  ASID STATE
---------------- -- -------- -------- -------- ---- ---------
SIGPATH_0A00529C 0A 000A015C JD0      XCFAS    0006 ACTIVE
SIGPATH_0C0052A3 0C 000C0160 JG0      XCFAS    0006 ACTIVE
SIGPATH_0D0052A0 0D 000D0138 JH0      XCFAS    0006 ACTIVE
SIGPATH_0100529B 01 000101E8 J90      XCFAS    0006 ACTIVE
SIGPATH_0200529E 02 000201EE JF0      XCFAS    0006 ACTIVE
SIGPATH_0300529F 03 00030247 JC0      XCFAS    0006 ACTIVE
SIGPATH_04005298 04 00040207 JB0      XCFAS    0006 ACTIVE
SIGPATH_050052A1 05 0005021D JA0      XCFAS    0006 ACTIVE
SIGPATH_060052A4 06 000601F9 Z0       XCFAS    0006 ACTIVE
SIGPATH_0700529A 07 00070215 JE0      XCFAS    0006 ACTIVE
SIGPATH_080052A2 08 000801FD TPN      XCFAS    0006 ACTIVE
SIGPATH_090052A6 09 00090208 J80      XCFAS    0006 ACTIVE
SIGPATH_1100529D 11 00110125 JI0      XCFAS    0006 ACTIVE
SIGPATH_12005299 12 00120112 JJ0      XCFAS    0006 ACTIVE
SIGPATH_130052A5 13 00130138 JL0      XCFAS    0006 ACTIVE

DIAGNOSTIC INFORMATION:  STRNUM: 000001B3 STRSEQ: 00000000
                          MANAGER SYSTEM ID:  0D0052A0
  NAME/MGR  #QUEUED   1STQESN   LASTQESN  CMPESN    NOTIFYESN
  J80       00000000  00000000  00000000  00000000  00000000

EVENT MANAGEMENT: MESSAGE-BASED       MANAGER SYSTEM NAME:  JH0
                                      MANAGEMENT LEVEL   :  01052010
```

If we displayed a structure that supports encryption but does not specify anything for **ENCRYPT** in the policy (or if you specified **ENCRYPT : NO**), you will still see that the **ACTIVE STRUCTURE** section reports **ENCRYPTION LEVEL: NOT ENCRYPTED.** Note that for this IXCPLEX1_PATH1 structure display, we've abbreviated the output for IXC360I by removing the connection details:

```
D XCF,STR,STRNM=IXCPLEX_PATH1
IXC360I  15.19.28  DISPLAY XCF 155
STRNAME: IXCPLEX_PATH1
 STATUS: ALLOCATED
 EVENT MANAGEMENT: POLICY-BASED
 TYPE: LIST
 POLICY INFORMATION:
  POLICY SIZE    : 2723840 K
  POLICY INITSIZE: 1369088 K
  POLICY MINSIZE : 1026816 K
  FULLTHRESHOLD  : 90
  ALLOWAUTOALT   : YES
  REBUILD PERCENT: N/A
  DUPLEX         : DISABLED
  ALLOWREALLOCATE: YES
  PREFERENCE LIST: CF4      CF3      CF2      CF1
  ENFORCEORDER   : NO
  EXCLUSION LIST IS EMPTY

 ACTIVE STRUCTURE
 ----------------
  ALLOCATION TIME: 08/07/2017 08:32:12
  CFNAME         : CF4
  COUPLING FACILITY: 002964.IBM.02.0000000840F7
                     PARTITION: 14   CPCID: 00
  STORAGE CONFIGURATION     ALLOCATED          MAXIMUM      %
   ACTUAL SIZE:                   1339 M           2660 M   50

   SPACE USAGE     IN-USE       TOTAL    %
    ENTRIES:            1      315507    0
    ELEMENTS:          30      315494    0

  PHYSICAL VERSION: D2F3A2A1 0C387BA7
  LOGICAL  VERSION: D2F3A2A1 0C387BA7
  SYSTEM-MANAGED PROCESS LEVEL: NOT APPLICABLE
  DISPOSITION    : DELETE
  ACCESS TIME    : 0
  MAX CONNECTIONS: 32
  # CONNECTIONS  : 15
  ENCRYPTION LEVEL: NOT ENCRYPTED
```

Similarly, if we display the ISGLOCK structure which does not support encryption since it's a Lock structure, you see that the ACTIVE STRUCTURE reports **NOT APPLICABLE - NOT SUPPORTED FOR STRUCTURE TYPE**. Note that for this ISGLOCK structure display, we've abbreviated the output for IXC360I to show only the ACTIVE STRUCTURE section:

```
IXC360I  14.10.17  DISPLAY XCF 115
STRNAME: ISGLOCK
 STATUS: ALLOCATED
.
.

ACTIVE STRUCTURE
----------------
 ALLOCATION TIME: 08/01/2017 01:50:58
 CFNAME         : CF4
 COUPLING FACILITY: 002964.IBM.02.0000000840F7
                    PARTITION: 14   CPCID: 00
 STORAGE CONFIGURATION     ALLOCATED          MAXIMUM      %
  ACTUAL SIZE:                    65 M             65 M  100

  SPACE USAGE                  TOTAL
   LOCKS:                     8388608

 PHYSICAL VERSION: D2EBBDC1 3A915CAD
 LOGICAL  VERSION: D2EBBDC1 3A915CAD
 SYSTEM-MANAGED PROCESS LEVEL: 8
 XCF GRPNAME    : IXCLO0B0
 DISPOSITION    : DELETE
 ACCESS TIME    : 0
 MAX CONNECTIONS: 32
 # CONNECTIONS  : 15
 ENCRYPTION LEVEL: NOT APPLICABLE - NOT SUPPORTED FOR STRUCTURE TYPE
```

RMF Monitor III Coupling Facility Activity panel now includes a new column **E** which will show whether the structure was encrypted (with a value **Y**), not encrypted (with a value **N**) or not applicable for encryption (with value of **-**) in the monitored interval being displayed.

```
                    RMF V2R3   CF Activity      - UTCPLXJ8

Samples: 60      Systems: 15   Date: 12/02/17  Time: 15.10.00  Range: 60    Sec


CF: ALL          Type  ST E System   CF    --- Sync ---  -------- Async -------
                                      Util   Rate   Avg    Rate   Avg   Chg  Del
Structure Name                         %            Serv          Serv   %    %
.
.
ISGLOCK          LOCK  A  - *ALL      0.6   2932     11    0.0      0  0.0  0.0
.
.
IXCPLEX_PATH1    LIST  A  N *ALL      1.2    0.0      0  267.9    118  0.0  0.0
IXCPLEX_PATH2    LIST  A  N *ALL     17.1    0.0      0  29207     33  0.0  0.0
IXCPLEX_PATH3    LIST  A  N *ALL     39.6    0.0      0  11005     44  0.0  0.0
IXCPLEX_PATH4    LIST  A  Y *ALL      2.5    0.0      0   3581     39  0.0  0.0
IXCPLEX_PATH5    LIST  A  Y *ALL     42.9    0.0      0  15975     36  0.0  0.0
IXCPLEX_PATH6    LIST  A  Y *ALL     45.8    0.0      0  16356     38  0.0  0.0
.
.
```

The RMF CF Activity post processor report has also been updated to indicate Encryption state for structures. In the GENERAL STRUCTURE SUMMARY section for the **COUPLING FACILITY USAGE SUMMARY** report section, a new **ENC** column is added to show whether the structure was encrypted, not encrypted or N/A (not applicable) for encryption, in the interval of time that this CF Activity report was created for.

```
                              C O U P L I N G   F A C I L I T Y   A C T I V I T Y
                                                                                                    PAGE    1
    z/OS V2R3           SYSPLEX UTCPLXJ8         DATE 09/11/2017          INTERVAL 001.00.000
                    CONVERTED TO z/OS V2R3 RMF   TIME 10.30.00            CYCLE 01.000 SECONDS

-----------------------------------------------------------------------------------------------------------------
 COUPLING FACILITY NAME = CF4
 TOTAL SAMPLES(AVG) =    59   (MAX) =    59   (MIN) =    58
-----------------------------------------------------------------------------------------------------------------
                                         COUPLING  FACILITY  USAGE  SUMMARY
-----------------------------------------------------------------------------------------------------------------
 GENERAL STRUCTURE SUMMARY
-----------------------------------------------------------------------------------------------------------------

                                                   % OF           % OF  % OF    AVG     LST/DIR  DATA      LOCK     DIR REC/
       STRUCTURE                             ALLOC  CF       #     ALL   CF     REQ/    ENTRIES  ELEMENTS  ENTRIES  DIR REC
 TYPE  NAME              STATUS CHG  ENC     SIZE   STOR     REQ   REQ   UTIL   SEC     TOT/CUR  TOT/CUR   TOT/CUR  XI'S

       ISGLOCK           ACTIVE      N/A     65M    0.0   201305   2.0   0.5   3355.1      0         0       8389K     N/A
                                                                                           0         0        167K     N/A
       .
       .
       IXCPLEX_PATH1     ACTIVE      NO       1G    0.1    27806   0.3   0.5   463.43    316K      315K       N/A      N/A
                                                                                           1        16        N/A      N/A
       .
       .
       IXCPLEX_PATH6     ACTIVE      YES     94M    0.0    1371K  13.9  14.6   22856      20K       20K       N/A      N/A
                                                                                           1        40        N/A      N/A
```

Similarly, the **COUPLING FACILITY STRUCTURE ACTIVITY** section will also now display the **ENCRYPTED = state**, such as the following screen caption:

```
-------------------------------------------------------------------------------------------------
                                        COUPLING  FACILITY  STRUCTURE  ACTIVITY
-------------------------------------------------------------------------------------------------

 STRUCTURE NAME = IXCPLEX_PATH6    TYPE = LIST   STATUS = ACTIVE                 ENCRYPTED = YES
          # REQ     -------------- REQUESTS -------------   -------------- DELAYED REQUESTS -------------
 SYSTEM   TOTAL              #     % OF   -SERV TIME(MIC)-    REASON     #    % OF   ---- AVG TIME(MIC) -----
 NAME     AVG/SEC           REQ    ALL      AVG    STD_DEV              REQ    REQ    /DEL      STD_DEV   /ALL

 JA0        106K   SYNC       0    0.0      0.0      0.0     NO SCH     0    0.0    0.0          0.0    0.0
            1775   ASYNC   106K    7.8     50.1     34.9     PR WT      0    0.0    0.0          0.0    0.0
                   CHNGD      0    0.0   INCLUDED IN ASYNC   PR CMP     0    0.0    0.0          0.0    0.0
                   SUPPR      0    0.0                       DUMP       0    0.0    0.0          0.0    0.0

 JB0        308K   SYNC       0    0.0      0.0      0.0     NO SCH     8    0.0   22.9         17.1    0.0
            5133   ASYNC   308K   22.5     31.4    133.5     PR WT      0    0.0    0.0          0.0    0.0
                   CHNGD      0    0.0   INCLUDED IN ASYNC   PR CMP     0    0.0    0.0          0.0    0.0
                   SUPPR      0    0.0                       DUMP       0    0.0    0.0          0.0    0.0

 JC0       60087   SYNC       0    0.0      0.0      0.0     NO SCH     0    0.0    0.0          0.0    0.0
            1001   ASYNC    60K    4.4     29.2     33.2     PR WT      0    0.0    0.0          0.0    0.0
.
.
STRUCTURE NAME = ISGLOCK          TYPE = LOCK   STATUS = ACTIVE                 ENCRYPTED = N/A
         # REQ     -------------- REQUESTS -------------   -------------- DELAYED REQUESTS -------------
SYSTEM   TOTAL              #     % OF   -SERV TIME(MIC)-    REASON     #    % OF   ---- AVG TIME(MIC) -----   EXTERNAL REQUEST
NAME     AVG/SEC           REQ    ALL      AVG    STD_DEV              REQ    REQ    /DEL      STD_DEV   /ALL   CONTENTIONS

JA0      23282    SYNC     19K    9.4     27.7     15.5     NO SCH     0    0.0    0.0          0.0    0.0    REQ TOTAL      41K
         388.0    ASYNC   4327    2.1     44.9     32.9     PR WT      0    0.0    0.0          0.0    0.0    REQ DEFERRED   211
                  CHNGD      0    0.0   INCLUDED IN ASYNC   PR CMP     0    0.0    0.0          0.0    0.0    -CONT          211
                  SUPPR      0    0.0                                                                        -FALSE CONT    130

JB0      40997    SYNC     41K   20.4      5.0      4.5     NO SCH     0    0.0    0.0          0.0    0.0    REQ TOTAL      54K
         683.3    ASYNC      0    0.0      0.0      0.0     PR WT      0    0.0    0.0          0.0    0.0    REQ DEFERRED   224
                  CHNGD      0    0.0   INCLUDED IN ASYNC   PR CMP     0    0.0    0.0          0.0    0.0    -CONT          224
                  SUPPR      0    0.0                                                                        -FALSE CONT    100
```

To date, once we completed the steps to enable the CF structure data encryption and could confirm the encrypted states of our structures, the usage of structures with encrypted data has not presented any challenges for us. We have done testing with recovery of structure connectors, ICSF, z/OS and CFs as well.

Pervasive Encryption: Middleware

V2R3 Data Set Encryption & IBM MQ

This topic discusses our experience with IBM MQ V8, V9 and V9.0x with Pervasive Encryption while testing z/OS V2R3 and z14 hardware.

To test in our environment, we defined an SMS data class with the Extended Attribute enabled as well as a Data Set Key Label to encrypt our new BSDS and archive logs. The new bootstrap data set (BSDS) was created and we used Access Method Services **REPRO** command to copy the current BSDS into the new BSDS.

We also setup some of our QMGR archive logs to exploit **COMPACTION** using zEDC hardware along with data set encryption by defining a separate SMS data class to also specify **COMPRESSION = ZR**

To encrypt our coupling facility (CF) structures we updated our CFRM policy to specify **ENCRYPT(YES)** for some our administration and application structures. Once the CFRM policy was updated and activated we rebuilt these structures to pick up the changes.

Please reference the following MQ documentation in IBM Knowledge center, particularly the **z/OS Data Set Encryption** section:

https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.pro.doc/q004180_.htm

We tested IMS exploitation of Pervasive Encryption on z/OS V2R2 and z/OS V2R3 systems using an IMS V14 6-way data-sharing IMSplex.

No IMS product APARs or PTF's were needed to support Pervasive Encryption.
We tested with the following IMS data sets encrypted:

1. VSAM non-HALDB databases
2. VSAM HALDB databases
3. IMS online log data sets (OLDS)
4. IMS system log data sets (SLDS)
5. IMS image copy data sets
6. CQS structure recovery data sets (SRDS)

Note: The extended addressability attribute is not supported for IMS data sets. We encountered the following error and set extended addressability attribute to N in data class to disable extended addressability:

IIEC031I D37-04,IFG0554P,IMS8,IMS8,DFSOLP00,7D73,SUBS9A,DBS8.IMS8.OLP0

We tested with following IMS CF structures encrypted:

1. IMS OSAM cache structure
2. IMS VSAM cache structure
3. IMS Full Function MSGQ structure
4. IMS Fast Path MSGQ structure
5. IMS virtual storage option (VSO) cache structures

For additional details of Pervasive Encryption and IMS support of Pervasive Encryption, please reference **Data Set Encryption for IBM® z/OS® V2.2 Frequently Asked Questions:**
https://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FQ131494

For **IMS Data Set Encryption:**
https://www.ibm.com/support/knowledgecenter/en/SSEPH2_15.1.0/com.ibm.ims15.doc.sag/system_admin/ims_dataset_encryption.htm

V2R2 + V2R3 Data Set Encryption & IBM CICS

This topic discusses our experience with IBM CICS TS 5.3, z/OS V2R2 & V2R3, along with z14 hardware.

To test in our environment, we defined several SMS data classes to accommodate a varied set of database files using either encryption or encryption and compression:

- VSAM RLS data sets for CICS OLTP workloads
- VSAM non-RLS data sets for CICS OLTP workloads
- VSAM RLS data sets for CICS VSAM batch workloads
- VSAM non-RLS data sets for CICS VSAM batch workloads

We implemented encryption on structures for RLSCACHE and CICS servers (temp storage, named counters, CF data tables) via the **ENCRYPT(YES)** CFRM policy parameter.

We implemented encryption on CICS system log and VSAM forward recovery data sets.

We chose to also encrypt our CICS system data sets, even if all of these may not contain sensitive data in a production environment: **DFHBRNSF, DFHDPFMB, DFHGCD, DFHINTRA, DFHLCD, DFHLRQ, DFHPIDIR, DFHTEMP, FILEA**

For CICS Data Set Encryption:
https://www.ibm.com/support/knowledgecenter/SSGMCP_5.4.0/configuring/cics/data-set-encryption-process.html

Pervasive Encryption: Useful Tooling

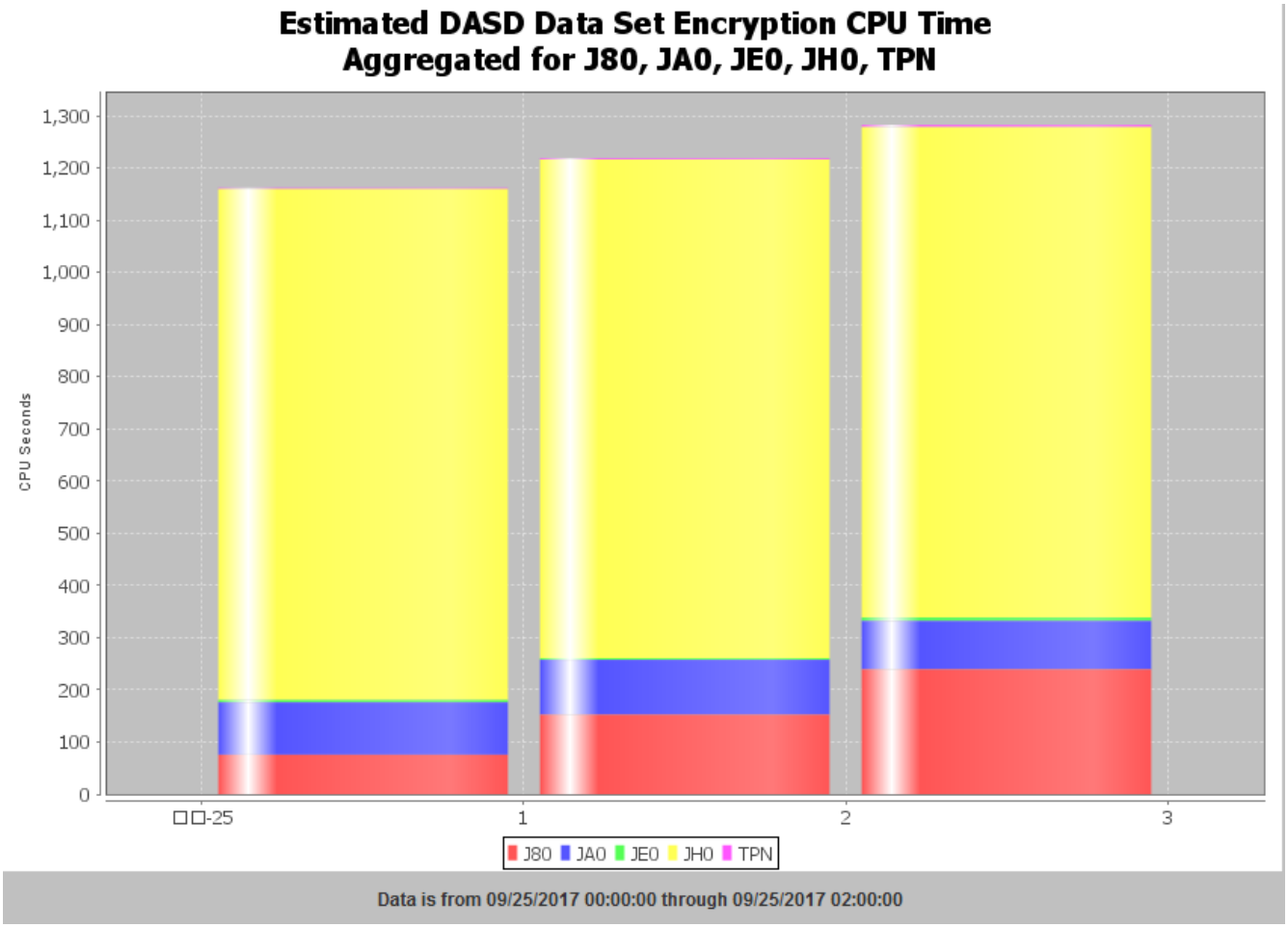
IBM zBNA for Pervasive Encryption Estimation

IBM zBNA is one of the capacity planning tools that can be downloaded from IBM PartnerWorld: http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS5132

A feature of zBNA included in Version 1.8.1 is the ability to evaluate a Z Server's data sets and CF structures for encryption capabilities. zPET used zBNA to identify encryption candidates on z/OS V2R3.

To accomplish these tasks, we followed the zBNA's User's Guide to collect SMF data as the input for the CP3KEXTR program running on z/OS. This produces *.edf* and *.dat* files for zBNA consumption.

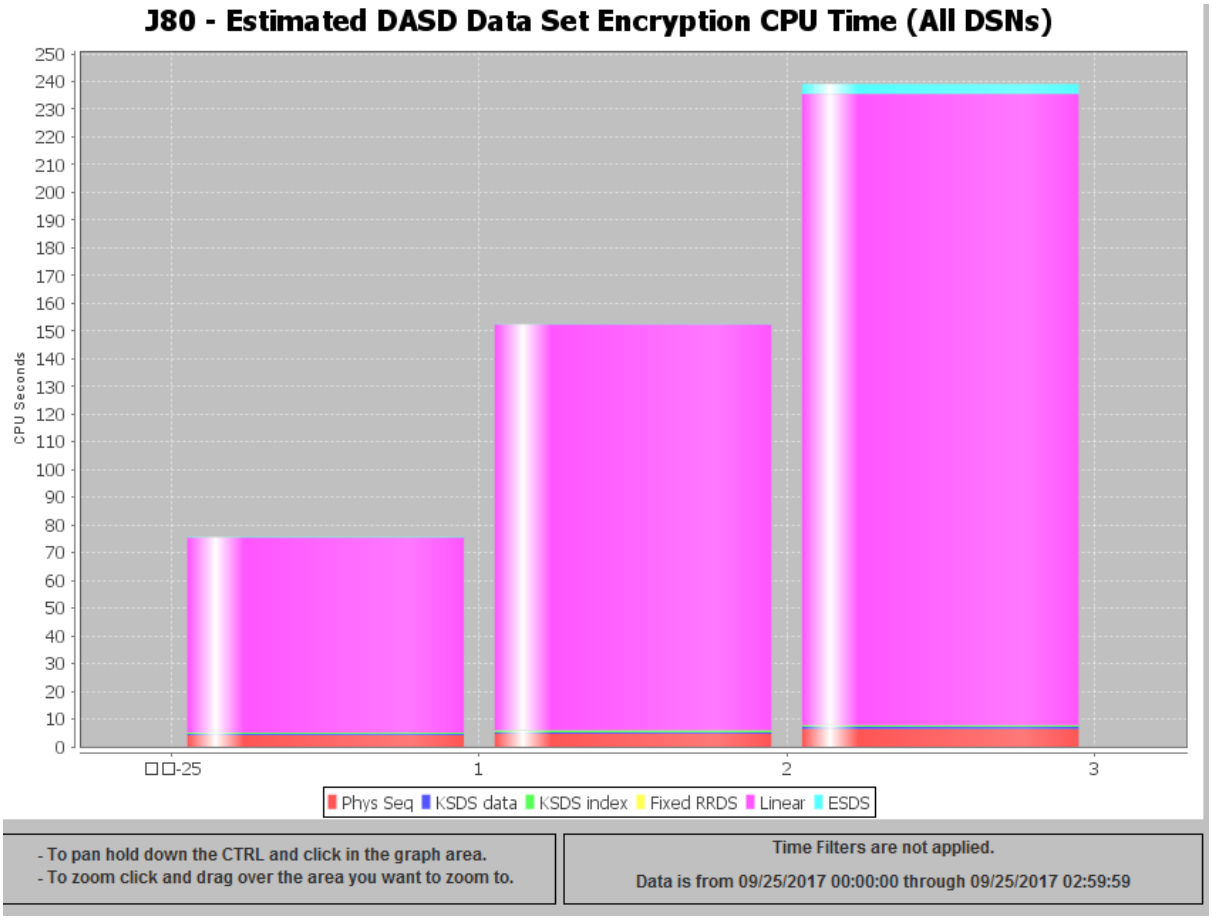
From zBNA program's main panel, we used the Encryption support via the Applications, Encryption menu to see an aggregated view of estimated DASD Data Set Encryption. The following graphic demonstrates the estimations over the course of three monitored RMF intervals from one of our CPC's and the five systems that run on it:

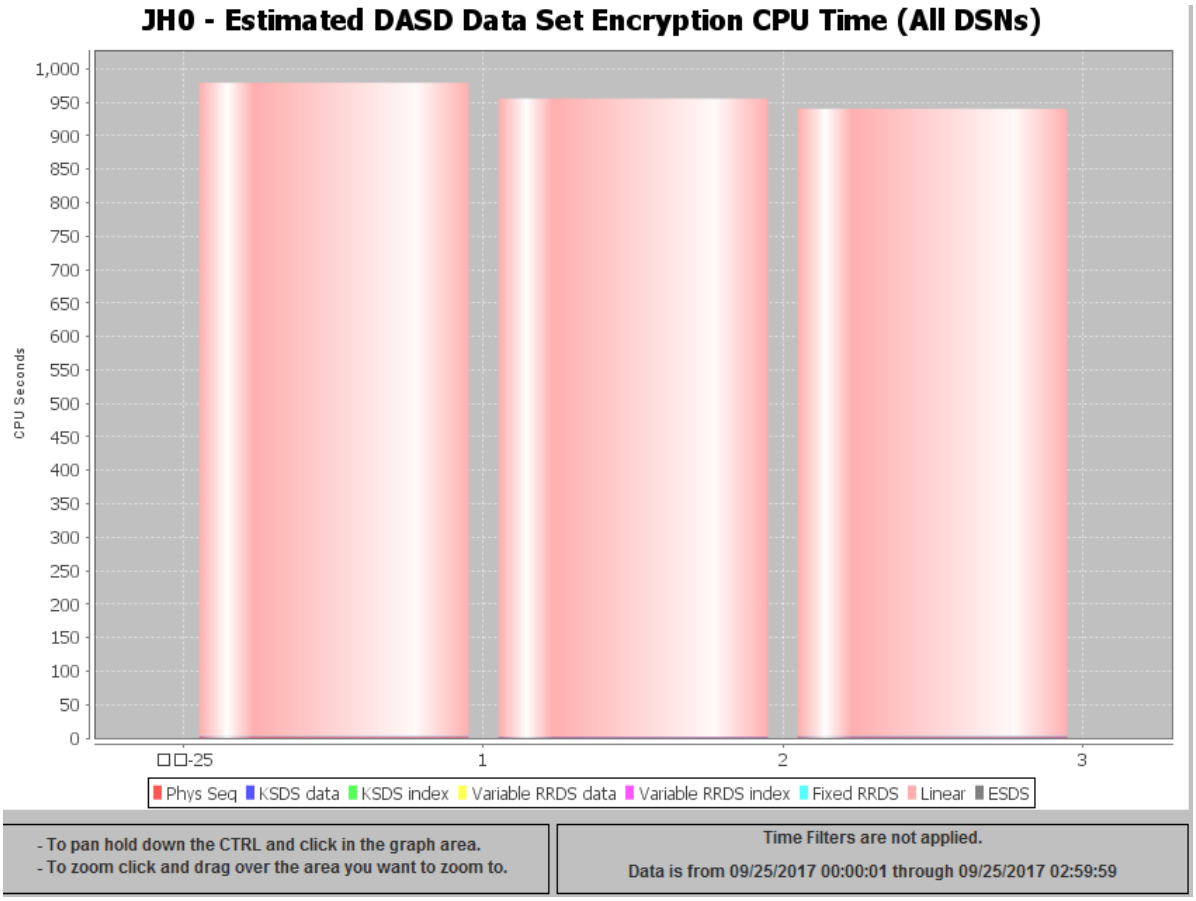


As shown in the visualized data, estimated CPU seconds for encrypting DASD data varies on different systems and in the different time intervals. System JH0 is estimated to have a higher CPU Time cost for data set encryption in the three monitored intervals than the other four systems monitored here.

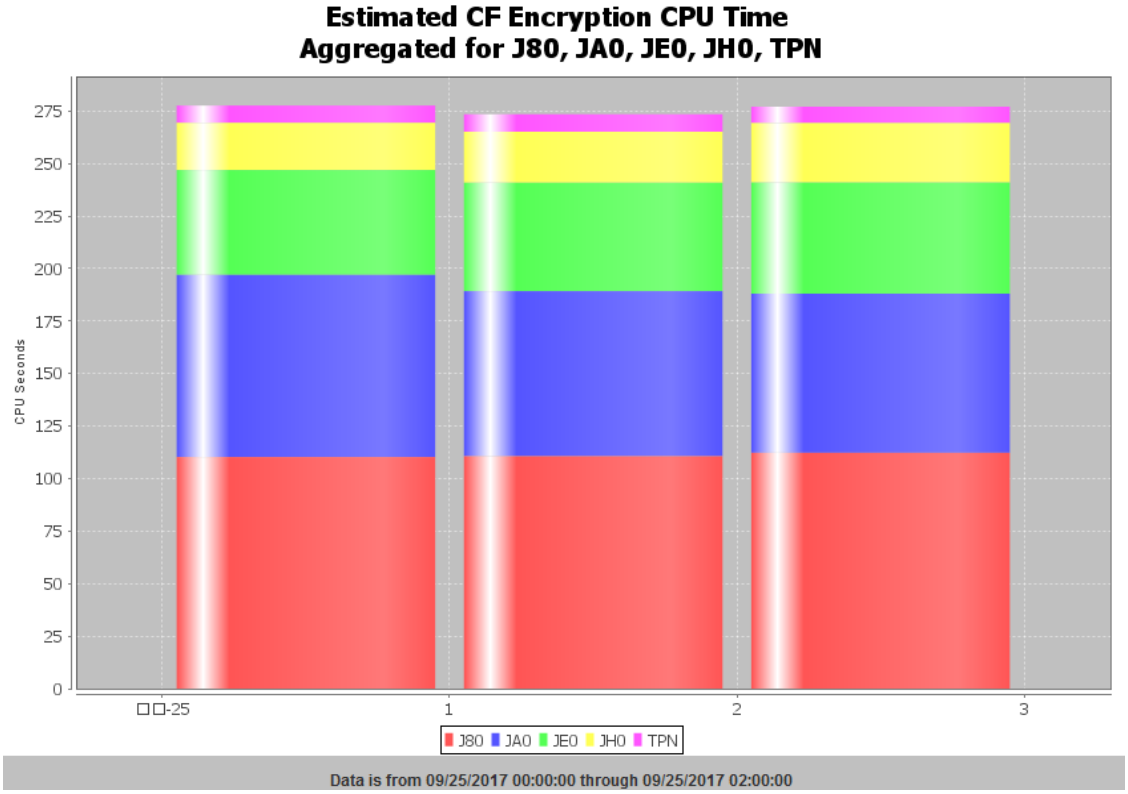
System J80 encryption estimates are more than three times higher in the third interval than in the first. This is because within this sysplex, there are different workloads running on different systems at different times which results in variations in data being accessed.

The following two figures compares two of these systems, and shows CPU time for encrypting all data sets which includes already encrypted data. It also includes estimated encrypted data further highlighting the variation of encryption CPU Time on system J80 while showing how consistent it is on system JH0:



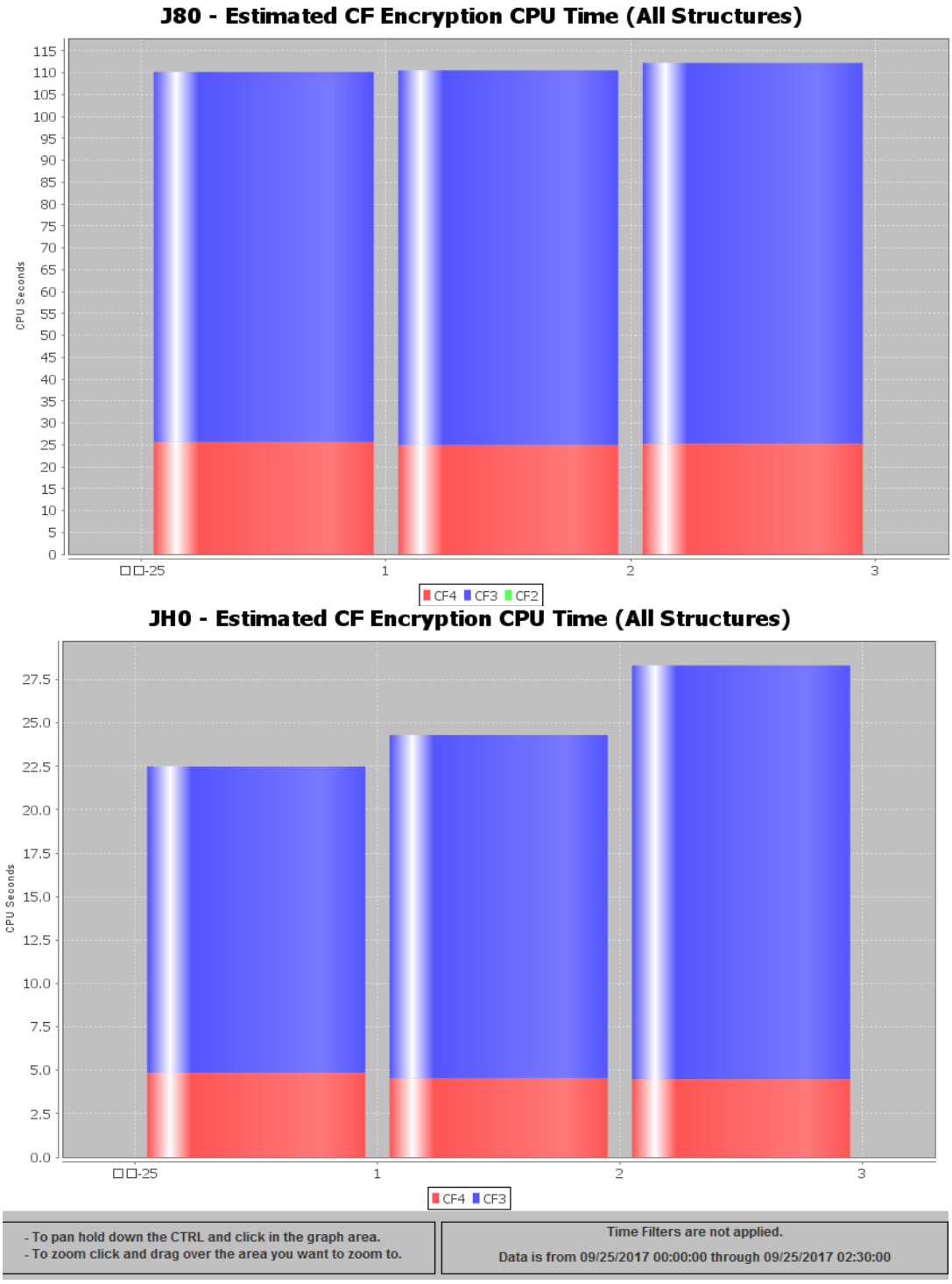


Following figure shows CF structure data encryption estimation from the same systems:



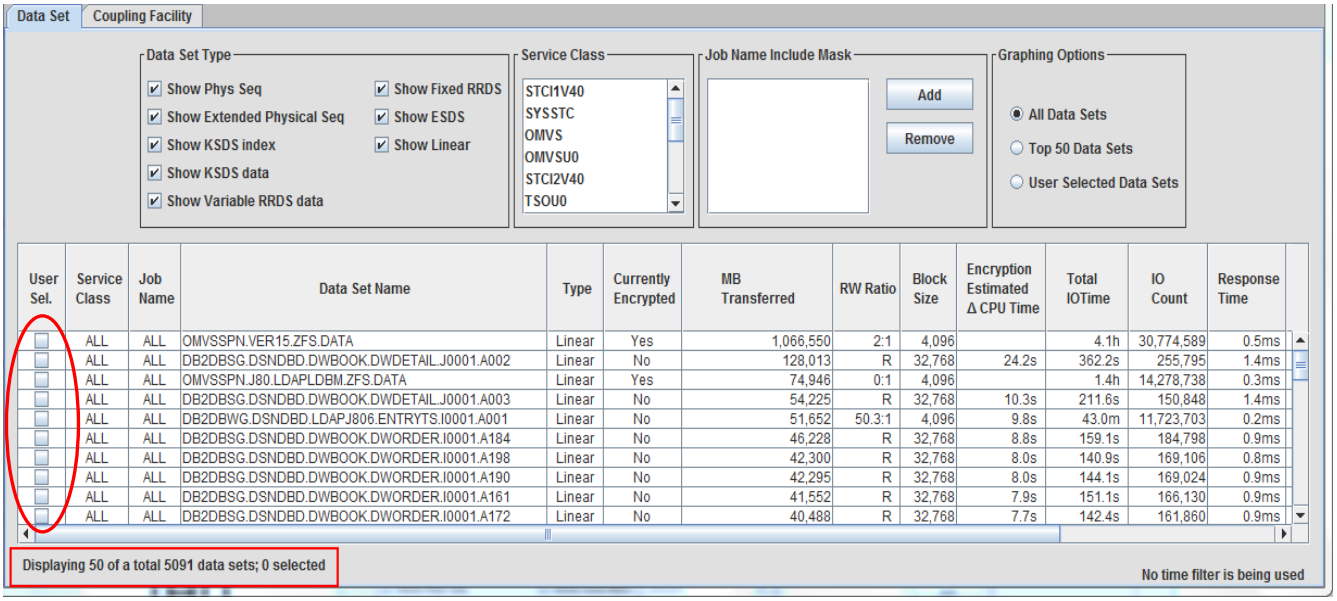
Like the previous data set encryption example, within one sysplex, the CF structure data estimated encryption results vary due to the workload variability between different systems.

The following two figures show two systems that include all connected CF structures:

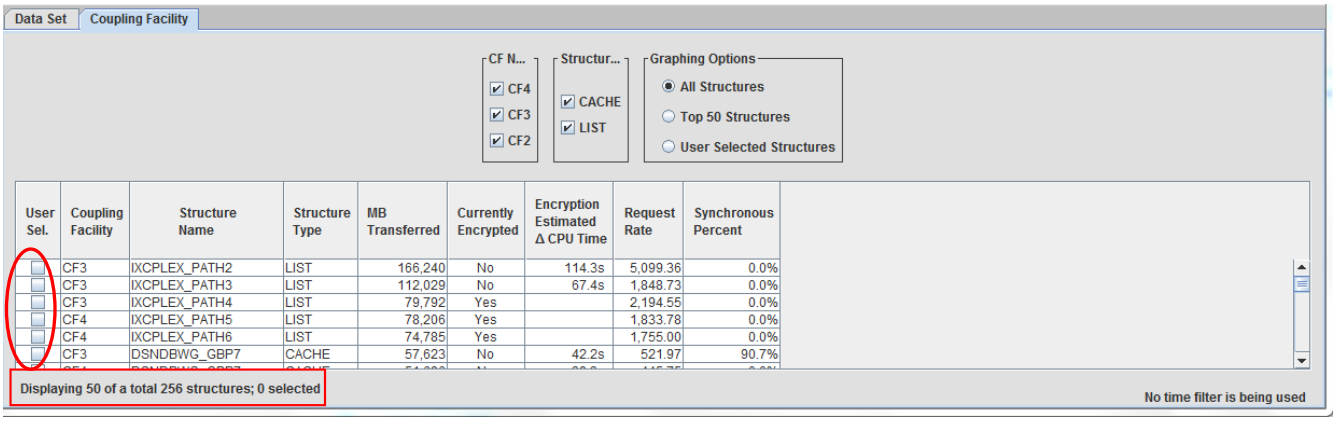


Another feature of zBNA is that it can enable individual data set selection for the user. With this feature, it is possible to include data sets that are either currently encrypted, or not, for evaluation. Other selectable options include **Encrypted Estimated CPU time** for individual data sets that are not currently encrypted.

Below is an example from one of our zPET systems:



For CF structures, a similar user selection view can be obtained as seen below:



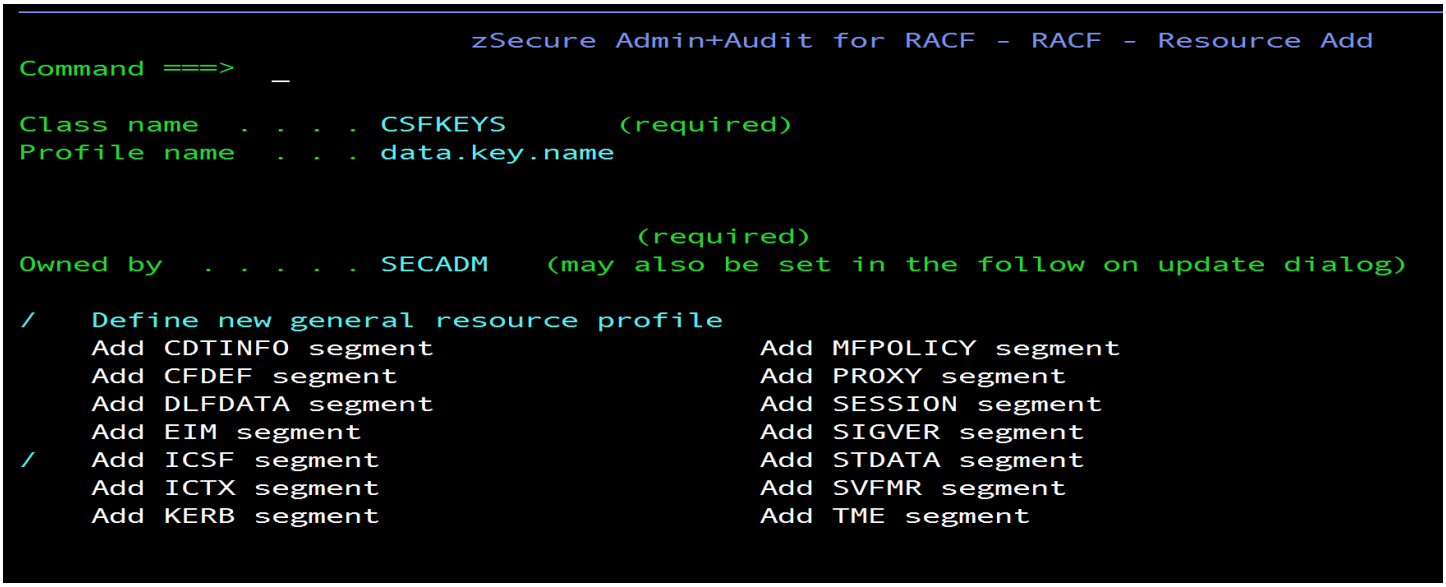
The user can obtain the output via exporting it to HTML or CSV for easy evaluation and usage. For more information and details, reference the zBNA User's guide.

IBM zSecure V2R3 contains features to support Pervasive Encryption. You can use zSecure to do many things such as:

- Protect and grant access to an ICSF Data Key
- Specify the key to use for encrypting a data set
- Run reports on keys and the data sets that are encrypted
- Provide Pervasive Encryption related SMF fields to QRADAR
- Include Pervasive Encryption related SMF fields in Event reports

The screenshot below shows the zSecure Admin option (**RA.R**) for adding a new general resource profile.

In this case, we are protecting an ICSF AES DATA key with a label of data.key.name by creating a profile in the CSFKEYS RACF class. By selecting the additional option for **Add ICSF segment**, it will allow us to specify additional information needed for this profile.



The following screenshot shows the information for the ICSF segment of the CSFKEYS profile. In the “key attributes” section, we changed the values for **Symmetric key CPACF wrap** and **Symmetric key CPACF return** to **yes** before hitting ENTER and proceeding through the panels to create the new profile.

```
zSecure Admin+Audit for RACF xCSFKEY ICSF segments
Command ===>                                                Scroll===> CSR
Class CSFKEYS, key data.key.name                 16 Jan 2018 12:48


  Identification                                                    PLEX1
  Profile name                  DATA.KEY.NAME
  Class                         CSFKEYS

  Certificate labels


  PKDS labels


  Key attributes
  Asym. key usage HANDSHAKE     Yes
  Asym. key usage SECUREEXPORT  Yes
  Symmetric key exportable by   ANY
  Symmetric key CPACF wrap      yes
  Symmetric key CPACF return    yes _
```

The above zSecure panels do the equivalent of the following RACF commands:

RDEFINE CSFKEYS DATA.KEY.NAME OWNER(SECADM)
RALTER CSFKEYS DATA.KEY.NAME ICSF(SYMCPACFWRAP(YES) SYMCPACFRET(YES))
SETR RACLIST(CSFKEYS) REFRESH

After listing the DATA key in option **RA.R** then placing **s** next to the profile name, then an **i** for **Insert new permit** in the user section, we are shown the panel below. We have specified that we want to give the ID "Banker1" an access level of **READ** to the key. In the optional conditions, we have specified **criteria** for the **When class** field and **sms(dsencryption)** for the **When resource/profile** field. This will give banker1 access to the key, but only when performing Pervasive Encryption.

```
                          zSecure Admin+Audit for RACF - RACF - New permit
Command ===>

Profile to be changed
Class  . . . . . . . CSFKEYS
Profile name . . . . DATA.KEY.NAME



Permit to be added
User or group  . . . BANKER1
Access level . . . . READ

Optional conditions for the permit
When class . . . . . criteria
When resource/profile
sms(dsencryption)_
```

The equivalent RACF command would be:

PERMIT DATA.KEY.NAME ID(BANKER1) ACCESS(READ) CLASS(CSFKEYS) WHEN (CRITERIA(SMS)DSENCRYPTION)))

Followed by

SETR RACLIST(CSFKEYS) REFRESH

The same zSecure panels can be used to permit users to any needed ICSF services in the CSFSERV class.

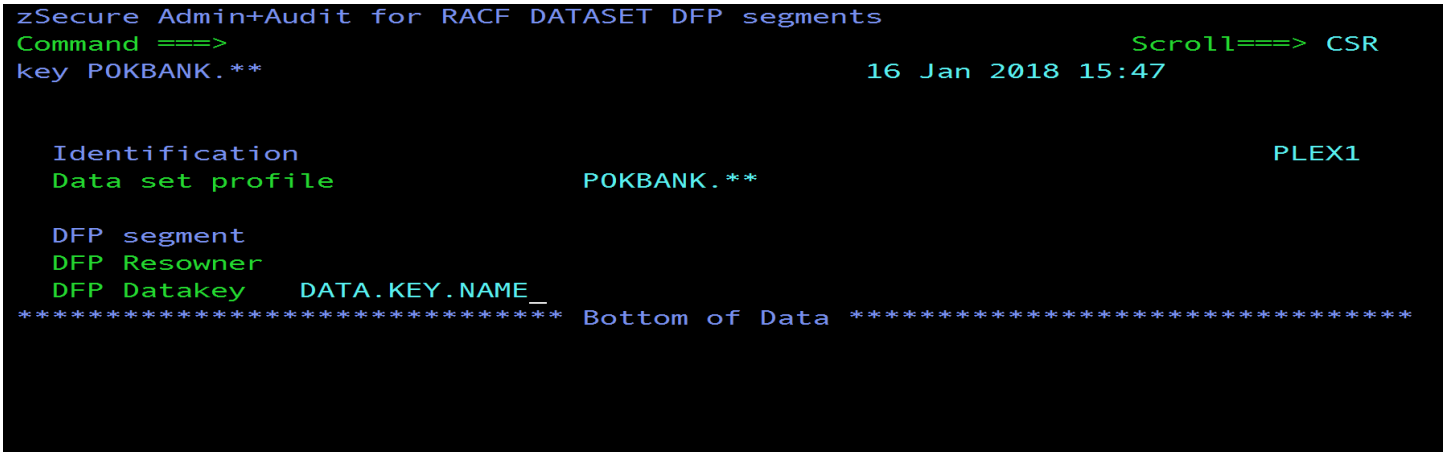
zSecure can also be used to add a DFP segment to a data set profile to associate a DATA key with the data sets that are to be encrypted. The panel below is displayed after listing a data set in **RA.D** and then selecting **Add new DATASET profile or segment**. We have unselected **Define new data set profile** and selected **Add DFP segment**.

```
                          zSecure Admin+Audit for RACF - RACF - Data set Add
Command ===>

Dataset profile . . POKBANK.**
Owned by  . . . . . SECADM    (may also be set in the follow on update dialog)

    Define new DATASET profile
/   Add DFP segment _
    Add TME segment
```

The screen below is then displayed, where we have specified the label of the DATA key to be used when encrypting data sets protected by this data set profile.



This issues the equivalent of the RACF command:
ALTDSD 'POKBANK.' DFP(NORESOWNER DATAKEY(DATA.KEY.NAME))**

In addition to the new panels that assist with the setup of Pervasive Encryption, zSecure also provides the means to report on Pervasive Encryption. The screenshot below shows option **RE.K** and the reports that are available.

If you are using the zSecure Admin task CKQRADAR to send SMF data to QRADAR, new SMF fields related to Pervasive Encryption are now being sent to QRADAR. These new SMF fields are also included in zSecure Audit's Event Reporting.

Helpful Documentation and Topics

Documentation for V2R2 and V2R3

- Data Set Encryption for IBM® z/OS® V2.2 Frequently Asked Questions:
  - https://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FQ131494
- Documentation Updates for APAR OA50569 z/OS Data Set Encryption z/OS V2R2:
  - http://publibz.boulder.ibm.com/zoslib/pdf/OA50569.pdf
- IBM KnowledgeCenter Pervasive Encryption for V2R3:
  - https://www-304.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r3izsp100/$file/izsp100_v2r3.pdf
- z/OS DFSMS Using the New Functions z/OS V2R3:
  - https://www-304.ibm.com/servers/resourcelink/svc00100.nsf/pages/zOSV2R3sc236857/$file/idak100_v2r3.pdf
- Pervasive Encryption - zOS Data Set Encryption:
  - https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W7df80301055d_495b_bb88_a0a2f84757c5/page/Pervasive%20Encryption%20-%20zOS%20Data%20Set%20Encryption

Conclusion and Retrospective

The importance of Pervasive Encryption to the IBM Z platform cannot be overstated. It is especially pertinent to businesses that depend on z/OS as their system of record given the critical nature of data that resides there. We feel that it will be a large factor in satisfying the security and protection standards for data for z/OS customers. If your experiences are similar to ours, we think it will be easily consumable and should have little to no impact to lines of business for z/OS customers.

There are many components, features and procedures that require action when implementing Pervasive Encryption. Furthermore, a strategic approach should be taken to plan for, implement and sustain your environment with Pervasive Encryption.

In terms of implementation, the team found that setting up Pervasive Encryption required working within various roles with specific skills to ensure accurate setup on the systems. Pervasive Encryption requires the appropriate hardware and compatible software to be available at time of setup. For Middleware, the team's consensus was that setup was simple enough for Middleware components, there was minimal setup required, and that Middleware could simply begin exploiting the encryption functions available with Pervasive Encryption once active. This made utilizing its functions quick and manageable.

Looking forward, it is important to recognize that threats will continue to exist as technology evolves. There is an unprecedented amount of data available in the world, and it will only continue to grow throughout time. The Pervasive Encryption umbrella will have to evolve and grow to meet the threats and the scale of the data. We plan to implement more in this umbrella and will continue to tell you about our experiences.

Acknowledgements

This document was compiled from the experiences of the entire zPET team, many of whom have created entries in our team's blog at http://ibm.biz/zPETBlog. We thank them for their dedication in providing quality testing and capturing their experiences. Without their individual contributions, our experiences would not be what they are today. We would therefore like to acknowledge:

- Alfred Lease Jr
- Antoine Saliba
- Bob Fantom
- Daniel Roth
- Dominic Rossillo
- Don Costello
- Evren O Baran
- Frederick Lates III
- Hui Wang
- Jing Ren
- Jing Wen Chen
- John J Corry
- Lisa Dodaro
- Lora Milczewski
- Mai Zeng
- Matthew Bosko
- Matthew Cousens
- Ming Qiao Shang Guan
- Phil Peters
- Ryan C Bartoe
- Thomas Sirc
- Torin Reilly
- Trent Balta
- Wei Song
- Xiao Chen Huang
- Xin Xin Dong
- Yu Mei Dai
- Zhao Yu Wang