Data Backups and Pervasive Encryption

A risk-based approach

Prepared by Lennie Dymoke-Bradshaw, Senior Security Consultant,
BMC Mainframe Services by RSM Partners

Table of Contents

Thanks are due to the following people for reviews and advice.

Mark Wilson, BMC Software

Chad Rikansrud, BMC Software

Andy Coulson, IBM UK Ltd

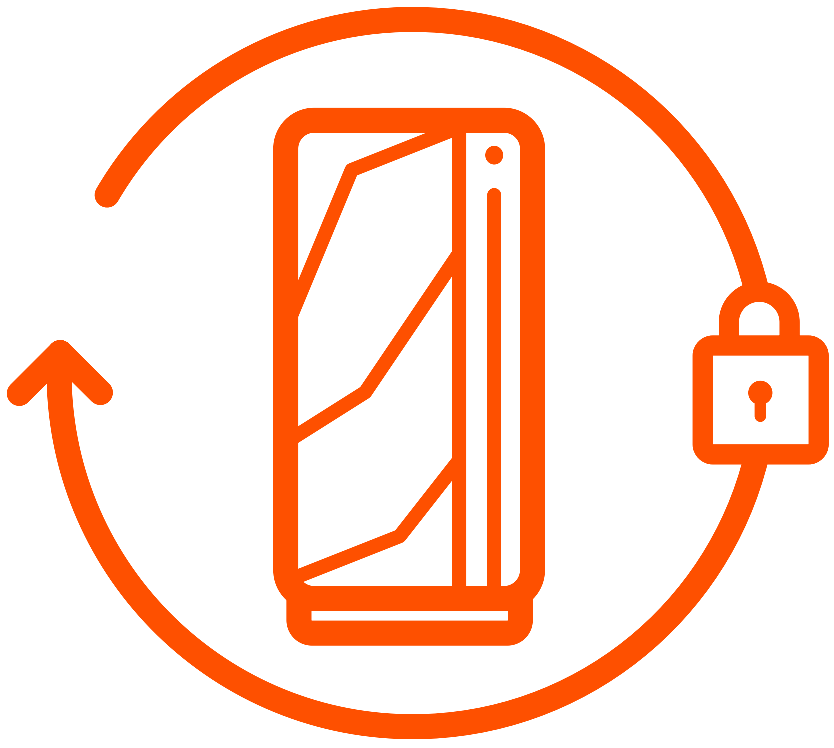
Introduction

This paper examines the reasons and methods for performing backups of data stored on IBM® z/OS® systems. It also examines how that reasoning and thinking may change with the introduction of z/OS pervasive encryption of data sets.

This document assumes the reader has familiarity with IBM®'s mainframe systems and the z/OS operating system. This includes methods and terms used on z/OS systems, such as "access method," "block copy," IBM® z/OS® DFSMShsm (HSM), peer-to-peer remote copy (PPRC), etc.

The paper does not assume a great deal of knowledge of encryption, merely that it serves to obfuscate data with the assistance of an encryption key, and that the same data can be made clear with the use of the same encryption key.

BMC Mainframe Services by RSM Partners recognizes that data on IBM disk and tape systems can be encrypted at device level rather than data set level. Such encryption makes use of IBM® Security Key Lifecycle Manager. This paper does not examine those capabilities. It concentrates of what is typically known as pervasive encryption of data sets.



Why do we take backup copies of data?

It may seem obvious: data backups are taken in case we lose or corrupt the primary version of the data. However, there are several different scenarios that backing up data protects us against, and we need to distinguish those situations from one another to ensure we are protected against each (every?) eventuality.

Corruption of data

Data can become corrupted. This can result from programming errors, hardware failures, malicious intent, process errors, and other causes. In order for the backup to be useful in the event of one of these situations, it may need to be physically separated from the master copy by some mechanism. For example, if the backup copy is for is protection against a disk failure, then it is essential that the copy is on a different physical disk or backup tape. If it is protection against process errors, then the backup should be a "point in time" backup and not a synchronous copy. A synchronous copy mechanism would replicate the error to the backup.

Logical loss of data

The data set containing the data can be deleted. This can happen accidentally, or, again, through malicious intent. The results are the same. As far as data recovery is concerned, similar rules apply. Separation of the backup copy from the master is important to ensure that whatever incident befalls the master the backup does not suffer the same fate.

Loss of a data center

Entire data centers can become either non-operational (e.g., through power loss) or can be destroyed (e.g., terrorist action, environmental disaster). In this situation, physical separation of the backup copy from the master is essential. However, it is especially useful if the copy is as current as possible, so synchronous copy mechanisms such as PPRC can be used to great effect.

A need to view data history

This is not really a requirement for a backup of data. This is a requirement for an archive of data. Archival requirements should be considered separately from backup requirements. However, it is inevitable that some archival requirements can be met by examination of historical backup copies of data.

Risks

As with all types of risk analysis, we need to consider both the likelihood of an event occurring and the impact if that event does occur.

Your analysis may indicate that a given scenario is extremely unlikely, but you may also consider that if it does occur, it may cause catastrophic damage to your business. In such a situation, you will want to ensure there is a defined way of recovery, even if that recovery is complex and time-consuming.

Other risks may relate to scenarios that would have a far smaller impact on your business but are nevertheless deemed more likely to occur. Your recovery from such risks needs to be far faster and almost certainly automated.

Summary

The important thing about backup copies of data is that data availability should not be subject to a single-event loss of the data. There should be some method of recovering the data following a single "unfortunate event." If the single unfortunate event is a disk loss, then placing the backup on that same disk will not protect you.

Methods of performing backups

Backup copies of data are performed using multiple mechanisms.

Access method copies

This is perhaps the simplest and easiest way to understand backup mechanisms. A simple program reads each record of the data set in turn and writes out another data set. The second data set can use a completely different name, can use a different block size (i.e., number of logical records per physical block), can be stored on a different volume, and can be stored on a different medium (e.g., tape as opposed to disk). In fact, it could be written over a network link to another system. If written on a tape or other such removable device, it can be subsequently moved to another location.

The crucial point that distinguishes this type of backup copy is that the program that reads the data uses a simple access method such as queued sequential access method (QSAM) or virtual storage access method (VSAM), frequently similar to that used by the programs that process the data for the application in question.

Block copy mechanisms

Block copy mechanisms use operating system interfaces at a lower level. They look at the disk device and copy the physical blocks of data to another place. That could be on the same physical device, or it could be to another device. The distinguishing property of this method of backup is that much of the access method processing is bypassed. This provides a far faster way of replicating the data.

Such methods are often used to take backup copies of entire volumes or application-specific data sets. Performance is the main advantage of this method of backup.

One of the most popular programs for block copy is IBM® z/OS® DFSMSdss (DSS). Other programs such as IBM® FDR® (Fast Dump Restore) are also available.

Real-time replication

Real-time replication of data also works at a low level. In this case, potentially even lower than the block copy mechanisms above. Each physical volume is paired with another volume on another disk array device that is usually at a separate location. The volumes are termed primary and secondary. Each I/O operation which results in a change to the primary copy is also performed on the secondary copy. In fact, the operation to change the data on the primary copy is not deemed complete until that same operation has completed on the secondary copy.

This type of copy is sometimes referred to as a synchronous copy. These types of copy are used for recovery from major events. The distinguishing property of these backups is that they can be correct down to sub-second times.

Methods like this are termed PPRC, Metro Mirror, Global Mirror, Global Copy, etc.

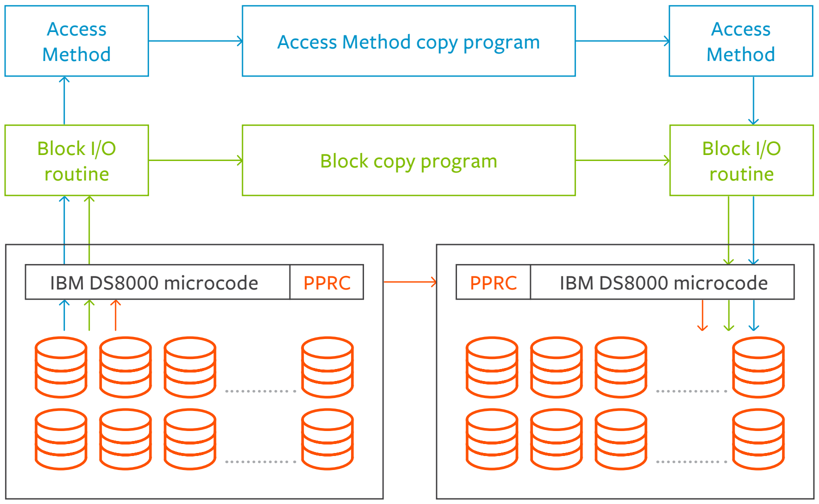


Figure 1: Copy mechanisms

Flash copy

This mechanism creates a logical table of the disk tracks that the data set occupies. Thus, two sets of the track map for the data exist. When an I/O operation for the primary copy takes place, the track is copied and then altered, and the corresponding track map is adjusted. However, the second track map still points to the original data and can be used to access the data as it was at the original "point in time" when the duplicate track table was created.

Flash copy mechanisms can be used to make very fast copies of data sets to the same physical volume. Once the flash copy has been made, it must be copied using other methods (usually a block copy method) to another physical volume. Once the copy has been made, the secondary track map can be discarded.

This method is particularly strong for making fast "point in time" copies with minimal interruption to active applications.

DFSMShsm

DFSMShsm (HSM for simplicity) is an IBM z/OS software component that monitors multiple disks and makes copies of data sets that have been changed. It can also be used to migrate data from one storage medium to another (i.e., disk to tape) on the basis of low usage.

When HSM performs its data-copy or data-move functions, it can be configured to use either an access-method-based copy or a block-copy mechanism. The choice is usually determined by the nature of the data set and the performance requirements. However, data sets in extended format are always managed by the block-copy mechanism (DFSMSdss).

Comparison of backup methods

Figure 1 illustrates two IBM® DS8000 storage arrays, each with many disks. A data set is to be copied from a volume on one of the DS8000s to a volume on the other.

If PPRC is used, then the backup is maintained synchronously using physical connections between the disk controller microcode in the two DS8000s. The data path is shown by the red arrows.

If block copy methods are used, then the program uses block I/O routines to read the data and pass it to the block I/O routine to write to the new data set. The data path is shown by the green arrows. If the access method copy is used, the program uses the access method to read each record and write to the new data set. The data path is shown by the blue arrows.

What requirements do we have of backups?

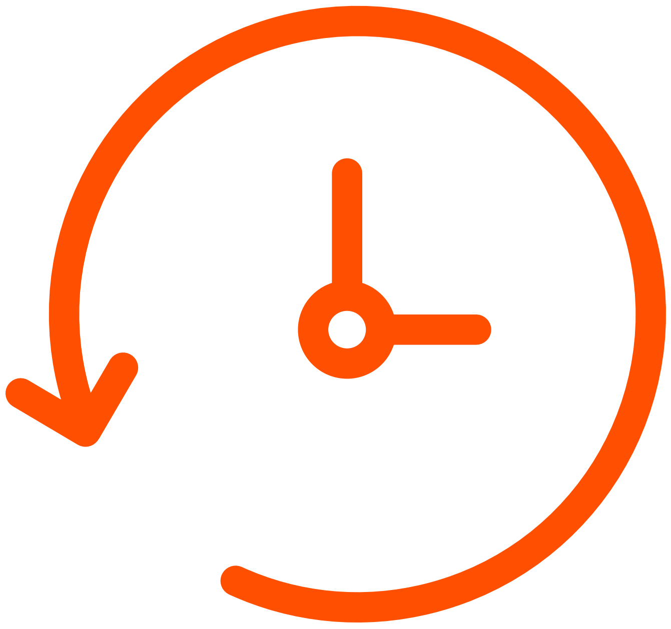
To be useful, backups should possess several qualities. Anything that affects these qualities can affect the ability to recover data from a backup, which affects the ability to recover the application requiring the data set.

Availability
For the backup to be used, it needs to exist and be accessible. If it is stored a long distance away, then it may have reduced availability if needed in an emergency. Nevertheless, in a data center disaster, such a backup is essential.

Integrity
The backup must be 100 percent accurate and the method of producing the backup must be resistant to being subverted.

Predictability
The backups must be taken at predictable times, or at least, known, times. If a sequence of backups exists, then the order in time is crucial.

Granularity
Backups for some applications and processes need to be granular. If a single data set is lost, then recovering an entire volume is going to be highly disruptive to any other application that has data on the same volume.



Types of media

This section examines the effects of types of media on the nature and value of backups. There are two main types: disk and tape. However, in many cases, physical tapes have been replaced by virtual tape servers (VTS).

Disk

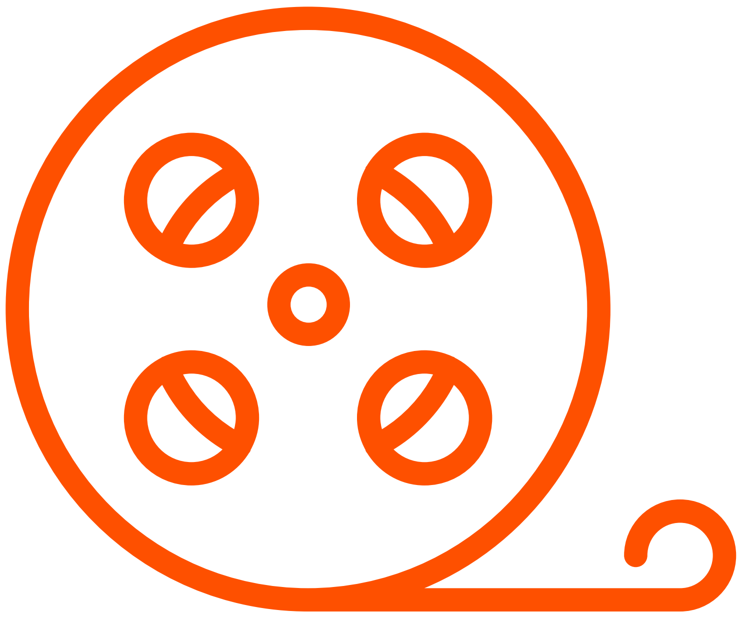
Disk systems on z/OS are frequently referred to as direct access storage devices (DASD). Modern disk systems make use of large arrays of individual disks modules similar to those used on smaller computer systems. There is a great deal of additional complexity associated with the disk systems. IBM's DS8000 systems and other DS8xxx systems make use of processors within the storage device to route data and manage multiple pathing from processors to disks.

Tape

Tape storage can supply access to large quantities of data sequentially. The nature of the medium prevents any type of random access and update capability. Individual tape cartridges are capable of being moved physically, and it is this attribute which has led to their use as backup systems, particularly where an offsite backup requirement is identified.

VTS and tape offload

Many modern systems make use of VTS devices. These use exactly the same software interfaces as physical tapes but store data on disk systems. Thus, the physical movement attribute is removed. These devices can move data to real tapes in a backend processor using a tiered data storage architecture. However, VTS devices can be placed remote from the processing capability, so the data is remote from the processor. As the VTS devices can be installed without changes to applications, they have proved popular for making data backup copies.
In general, and for the purposes of this paper, the use of VTS devices is deemed equivalent to the use of physical tape devices.



Risk table

This table presents various risks and examines how the different ways of data backup can mitigate them.

Threat	Access method to disk	Access method to tape/VTS	Block copy to disk	Block copy to tape/VTS	Synchronous
File corruption	Yes	Yes	Yes	Yes	No
Malicious corruption	Yes	Yes	Yes	Yes	No
Accidental data set deletion	Yes	Yes	Yes	Yes	No
Disk failure	Yes	Yes	Yes	Yes	Yes
Data center loss	Yes, if copy is offsite	Yes, if copy is offsite	Yes, if copy is offsite	Yes, if copy is offsite	Yes

Table 1 Threats and mitigations by backup method

In the table, a value of "Yes" shows we are protecting against the risk in the left-hand column. "No" shows we are not protecting against the risk.

Different methods of data backup clearly provide mitigation against different types of threats.

In particular, there is little difference between an access-method backup and a block-copy backup.

In addition, the medium used for data storage appears largely irrelevant.

Pervasive encryption of data sets

IBM has introduced a technology that allows data sets to be encrypted. This means that the data stored on the disk cannot be readily understood by humans or by an application program without first passing through a decryption process. We can examine how this function works and what effects this has on decisions in data backup.

Data set encryption

The encryption and decryption process used for each data set uses a method called advanced encryption standard (AES). To encrypt each piece of data, the data is passed into the AES algorithm together with an encryption key. Each encryption key is 32 bytes in length. To decrypt the data, the same key must be used.

To keep the encryption keys safe, z/OS uses a software component called an integrated cryptographic support facility (ICSF) to store keys in a VSAM data set. However, before each key is stored, it is itself encrypted (or wrapped) using a "master key." The master key is stored inside a tamper-proof device that fits into the System z frame, which ensures it is never exposed. The device is called a Crypto Express card; it is a class of device known as a hardware storage module (HSM). These HSMs have their security levels assessed against a National Institute of Standards and Technology (NIST) standard called FIPS 140-2. There are four levels of security for this standard. The Crypto Express device meets the highest level (level 4) of this standard.

The storage management subsystem (SMS) has automatic class selection (ACS) routines to assign attributes to new data sets, which can also be specified in job control language (JCL) or supplied from profiles in the RACF database. Using those SMS ACS routines coupled with RACF profiles can greatly simplify the assignment of encryption keys to data sets. The chosen assignment of encryption key names to data sets is kept in the z/OS catalog. Thus, when there is a need to access a data set, the name of the required key is easily identified. RACF can provide distinct access to the key and the data set. By granting access to a data set but denying access to its encryption key, storage administrators can copy, move, and back up data sets without requiring any access to the underlying data contained within the data set.

So, in order to process a data set, we require access to its encryption key, in addition to the data set. If we have access to the data set but no access to the key, we cannot access the underlying data. The data set we have is useless. By destroying the key of a data set, we are effectively destroying the data set itself. Indeed, this can be a very fast method of destroying data, if that is a requirement. It can be cryptographically destroyed simply be destroying its encryption key.

The encryption of data (when writing data to a disk) and the decryption of data (when reading data from a disk) occurs in the access method (e.g., QSAM or VSAM). This means that in nearly all cases, applications can continue processing data on a disk without having to be modified. See Figure 2.

[1] This means that there can be 115,792,089,237,316,195,423,570,985,008,687,907,853,269,984,665,640,564,039,457,584,007,913,129,639,936 different keys. The chances of finding the right one by accident is pretty slim.

The access method identifies the name of the key that is needed by retrieving it from the catalog. It retrieves the key from ICSF in a special form (known as a protected key), and then makes use of special IBM® System z® machine instructions called message security assist (MSA) instructions to perform encryption or decryption.

These instructions are executed in the part of the System z processor known as the CPU assist for cryptographic functions (CPACF). The key is used without ever being stored in z/OS memory in clear form. Please refer to documentation of CPACF Protected-key processing for more information.

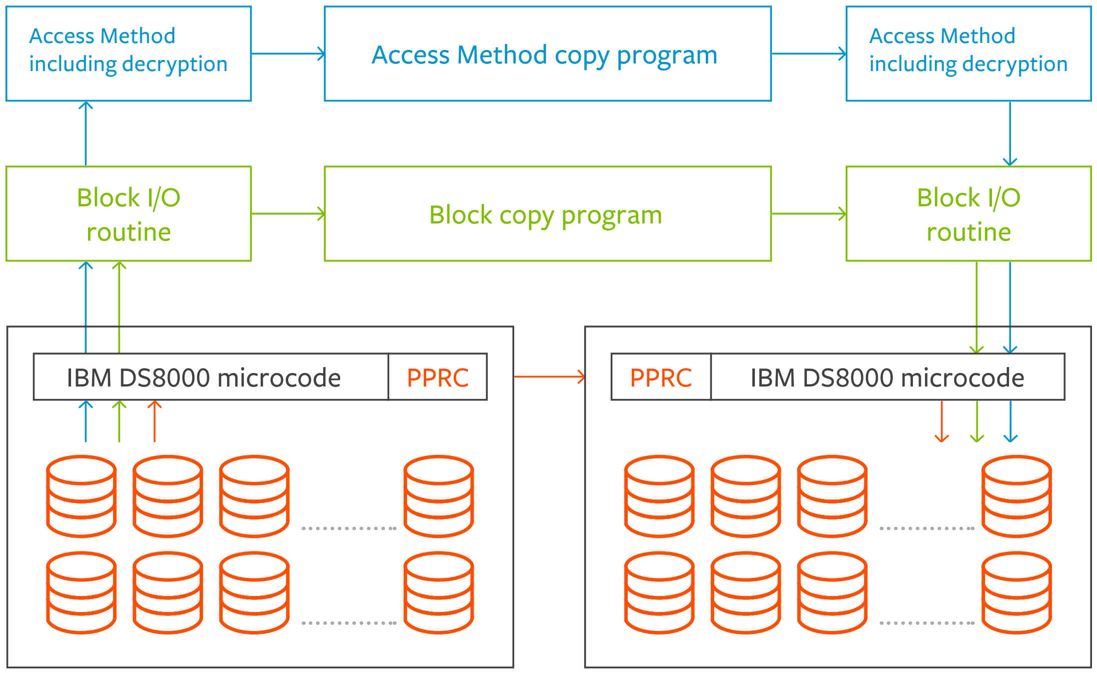


Figure 2: Data backup methods with encryption

Consequences of data set encryption

There are some important implications of encryption of data sets and of how this is performed in z/OS.

KEY LOSS[2] = DATA LOSS

If the encryption key to a data set is lost without hope of recovery, then all the data sets that were encrypted with that key have been lost.

Therefore, it is essential that the store of keys that are used for data set encryption is also regularly backed up. On z/OS systems, this key store is known as the cryptographic key data set (CKDS). It is a VSAM data set and access to it should be tightly controlled.

If data sets are rarely used, then the loss of the encryption key for that data set may not be noticed until an attempt is made to process the data set. If the keys were lost a long time ago, then it may not be possible to recover the key from backups of the CKDS.

Strict master key control is essential

The master keys used for the encryption of keys in the CKDS are normally changed on a periodic basis.[3] Some installations use a period of three years or two years, but more frequent changes such as once per year are more likely to be required by recent

[2] I have deliberately used capitals to describe this risk.

[3] ICSF uses four master keys. Two of those keys are used for the CKDS, the first for AES keys and the second for data encryption standard (DES) and triple DES keys.

security and audit standards. In addition, if there is any chance the master key has been exposed, those same standards normally require the key to be changed outside of the normal periodicity.

As each key in the CKDS is encrypted under the master key, we need the master key to access the encryption key; and as already stated, we need the encryption key to access the data. In other words, loss of the master key will also render the data protected by its wrapped keys completely lost.

Managing encryption keys is important

Maintaining access to the encryption keys is vitally important. Over time, large numbers of keys may build up in the CKDS. Establishing which keys are needed and which may be discarded can be a difficult and risky task.

ICSF allows keys to be placed in various states. They can be active, inactive or archived. They can also be deleted e.g., destroyed, discarded, demolished, eliminated, eradicated, wiped out, lost, not retrievable, gone, no more. If a key is not available, then the data encrypted with it is not available either. It, too, is effectively deleted e.g., destroyed, discarded, etc.

Option 5.5 in the ICSF panels is a browser facility for the CKDS that allows the movement of keys between states. It also allows the deletion of keys. Careful controls need to be placed on the use of these key browser facilities.

The periodicity of master key changes is important

Each master key in the Crypto Express device has three versions.

- **New master key.** This is a master key that has been entered into the Crypto Express but has not yet been activated.
- **Current master key.** This is the master key that is used for decrypting keys in the CKDS.
- **Old master key.** If a key name (or key label) is presented to ICSF and it is found not to be encrypted by the current master key, the old master key can be used by some applications. This usually applies in situations where keys are wrapped by the master key, but not held in the CKDS.

The three keys are stored in three registers. The old master key will be available only until the new master key is activated.

When the new master key is activated, the key in the current master key register replaces the key in the old master key register; and the key in the new master key register replaces the key in the current master key register. The original contents of the old master key register are discarded.

Using a batch program, the CKDS is then processed so that each key is decrypted using its old master key value and then re-encrypted using the new master key value. Each operation is carried out within the Crypto Express card so that no clear key is ever exposed.

Now consider the following situation:

Example 1

Let us assume the CKDS is backed up daily for a cycle of 60 days. Many people might consider that sufficient. But to be absolutely sure, we will keep monthly and annual backups as well. Let us suppose those go back seven years. Our system has a standard of changing the master key each year. We have a data set that is processed rarely. Let us say a new consolidation is produced only once per year but annual versions exist, corresponding to the last seven years. Each data set has its own encryption key. Requirements for processing the data sets arise from time to time, but there may be a delay of more than one year between those requirements. If a key for one of those (older) data sets is accidentally deleted today, the situation may not be discovered for two or three years. In that case, will we be able to recover the data?

In this example, it appears we will have a copy of the CKDS that should contain a copy of the key. However, the key will have been encrypted under a very old copy of the master key. Certainly, it will be neither the current master, nor the old master.

Even if we had a way of recovering the master key or storing old values of that key, performing the operation would be difficult. If it was possible, it would probably require separate hardware to perform.

The example above shows that the patterns of data usage and key management need to be aligned.

How we take backup copies is more important

If backup copies are taken using block copy methods such as DSS or FDR, then the data is copied without reference to the encryption keys. This has two effects.
The first is that access to that encryption key is not checked. So, if the key had been removed, this would not be discovered at that time. That is probably not so important, as the data set would be processed soon after using that key. If it were missing, this would soon be discovered.
The second effect is that the backup is now encrypted under the same key as the primary copy. This means the data is vulnerable to total loss caused by a single event: the destruction of the encryption key. This, too, may not be serious so long as methods are in place to recover individual keys from backup copies of the CKDS or somewhere else.

Example 2

An application uses a data set frequently. It is backed up on a daily cycle using DSS. A user with malicious intent corrupts the encryption key for the data set. The administrator for the application finds he cannot access the data and so recovers the data from the latest backup. He encounters exactly the same problem as with the master copy because it needs the same key. The data is entirely lost until the encryption key is recovered.

This second example shows how easy it is to move to a situation where a single piece of data corruption, i.e., of the encryption key, causes the loss not only of the primary copy of the data, but also of all the backups.

One alternative approach is to use access method data copy utilities and copy the data to new data set names which use different encryption keys. The chances of having at least one key available are higher.

It should be apparent that block-copy mechanisms will NOT achieve this level of key separation.

Disk vs. tape

IBM's current systems will handle the encryption of data sets on disk. It does not handle the encryption of data sets on tape. The same applies to data sets stored on VTS systems.

This means that if data is encrypted on disk and then copied to tape using an access-method copy, the data will NOT be encrypted on tape. However, if the data is copied to tape using block-copy mechanisms, the data will be encrypted on the tape.

Key naming standards

It is a good idea to consider using different keys for the backups of an application from those used for the primary data of the application. This will ensure that any access method backup remains available even if the key for the primary data is lost.

Given the ease with which keys are displayed and can be deleted in the CKDS browser, it might also be useful to use key naming standards to separate keys for primary data from those for backup data. Using this method, we might have a key naming standard such as,

```
DSET.PRIM.appname1.key001
DSET.BACK.appname1.key001
```

By having a second level qualifier of PRIM and BACK, the keys for the application will not normally be shown together, and so are less likely to be accidentally deleted together. At least this will be true if there are multiple applications using encryption, as shown below. The primary keys and backup keys for "appname1" are not consecutive in the list.

```
DSET.PRIM.appname1.key001
DSET.PRIM.appname2.key001
DSET.PRIM.appname2.key002
DSET.PRIM.appname3.key001
DSET.BACK.appname1.key001
```

Risk table

We can again look at ways of backing up data and how they mitigate risk but also take into account the encryption of that data. The results are shown in Table 2, which has significant differences from Table 1.

Now we must consider the effects of key loss, a situation where having backup data protected by a different key provides benefits. This requires that block backup methods and synchronous methods are avoided.

This risk table also separates backups-to-disk from backups-to-tape, as the encryption capabilities to tape are different.

Threat	Access method to disk	Access method to tape/VTS	Block copy to disk	Block copy to tape/VTS	Synchronous
File corruption	Yes	Yes	Yes	Yes	No
Malicious corruption	Yes	Yes	Yes	Yes	No
Accidental data set deletion	Yes	Yes	Yes	Yes	No
Disk failure	Yes	Yes	Yes	Yes	Yes
Data center loss	Yes, if copy is offsite	Yes, if copy is offsite	Yes, if copy is offsite	Yes, if copy is offsite	Yes
Key loss	Yes, if backup protected by different key	Yes (but backups not secured i.e. not encrypted)	No	No	No

Table 2. Threats and mitigations by backup method with encryption keys

In the table above, a value of "Yes" shows we are protecting against the risk in the left-hand column. "No" shows that we are not protecting against the risk.

In all cases, we are protecting against single events. A dual-event risk table would need a great deal more examination.

Key management and backup

The change in the risk table introduced by the use of encryption is entirely due to the additional risk case of the loss of data encryption keys. This can be mitigated by an external mechanism for backup and recovery of those keys. Such a mechanism could also provide for archiving of keys. In fact, a full management cycle of keys can be produced. This could include the original derivation of those keys and the deployment to systems that need the keys. That deployment could include target devices of other platforms as well as to z/OS sysplexes.

Enterprise key management foundation (EKMF)

An external mechanism for key management and backup exists. It was originally called a distributed key management system (DKMS) and was produced by IBM at its Crypto Competence Center in Copenhagen (CCCC). It is now marketed by IBM worldwide as EKMF). More details can be found here

Benefits of EKMF

The following information is largely extracted from the IBM website mentioned above.

Multiplatform, multisite, and multivendor support

EKMF provides the facility to perform all key and certificate management functions across different platforms, operation systems, and geographical locations, and for a variety of key end points. Specifically, EKMF currently supports the following cryptographic platforms:

- IBM® mainframe cryptographic co-processors (Crypto Express6, Crypto Express5, Crypto Express4) on z/OS
- IBM® 4765 and IBM® 4767 on Windows® and Linux® on Intel86 platforms
- IBM® 4765 and IBM® 4767 on Power Systems
- IBM® RACF® keystore and keyrings
- IBM® WebSphere® DataPower®
- Other vendor crypto hardware such as Thales®
- SSL key stores such Java Key stores, PKCS#12, and KDB

Central repository

All keys and certificates are stored in a central repository together with meta data such as activation dates and usage. By storing all key material in a central repository, backup is easily achieved by including the database in existing database backup procedures. This facilitates easy recovery if keys or certificates are lost.

Monitoring of keys and certificates

Expiration of key material is monitored, and alerts are generated in due time to initiate replacement. This is crucial for certificates as an expired certificate most often means that a service is unavailable.

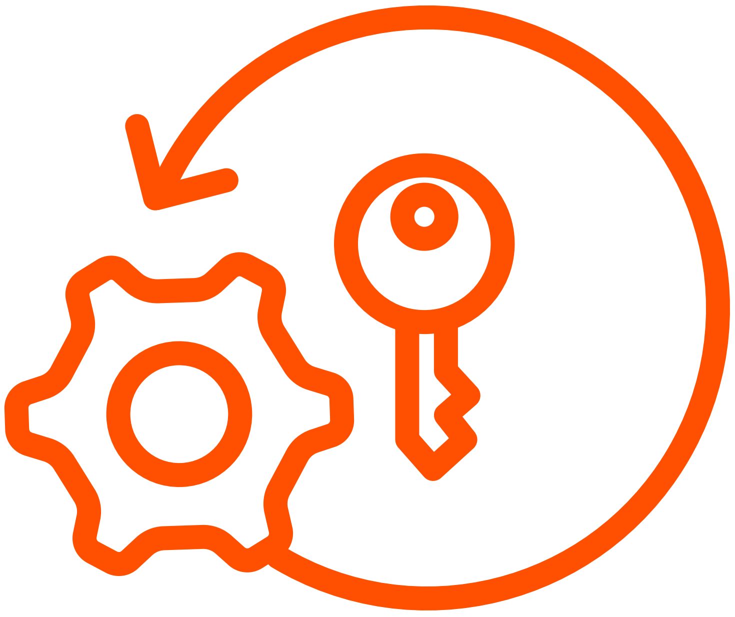
Security features

- **Secure key generation.** The security of the system is highly dependent on the method of key generation. In EKMF, key generation takes place within the IBM® 4765 cryptographic co-processor where a random generator generates the keys. RSA key generation is in conformance with ANSI 9.31.

- **Role-based access control.** The EKMF access control system is role-based and controls access to functions and keys. The system administrator can define which functions and keys are available for each user.
- **Dual control.** EKMF can be configured to require that two or more persons log on to activate EKMF, thus providing dual control for all operations.
- **Audit logging.** Every important activity is logged in a DB2 table and, if available, in IBM System Management Facilities (SMF).

Workflow

Effective working with high key volumes is provided via fully-automated and semi-automated processes and bulk key management.



Conclusions: next steps

Pervasive encryption of z/OS data sets is a strong strategy for addressing certain security and compliance requirements.

However, pervasive encryption for data sets does not protect you against all threats. It does not protect you from data exfiltration by malicious actors who gain access to your keys. If an attacker can gain access to the credentials of a user who has access to the keys themselves, or the RACF database or other security controls, then they will still be able to take exfiltrate your data.

When implementing pervasive encryption of z/OS data sets, we recommend the following action points to consider how data should be replicated for backup purposes.

1. Perform a risk assessment to see where the risks are for each application with encrypted data.
2. Develop a good understanding of how data set encryption works; this is vital to help design the optimum backup strategies.
3. Recognize that strategies currently used for the management of clear text data may require changes when managing encrypted data.
4. Build a good understanding of the usage patterns of application data sets so you can ensure keys are retained for use when needed.
5. Gain an appreciation of how the choice of backup methods affects the set of keys needed to recover data sets.
6. Formulate a strong strategy for taking backup copies of the CKDS.
7. Consider key-naming standards to help mitigate risks.
8. Enterprises should also give serious consideration to deploying EKMF to manage and secure data encryption keys.

For more information

To learn more about Data Backups and Pervasive Encryption, please visit bmc.com

About BMC

From core to cloud to edge, BMC delivers the software and services that enable over 10,000 global customers, including 84% of the Forbes Global 100, to thrive in their ongoing evolution to an Autonomous Digital Enterprise.

BMC – Run and Reinvent

www.bmc.com

522105