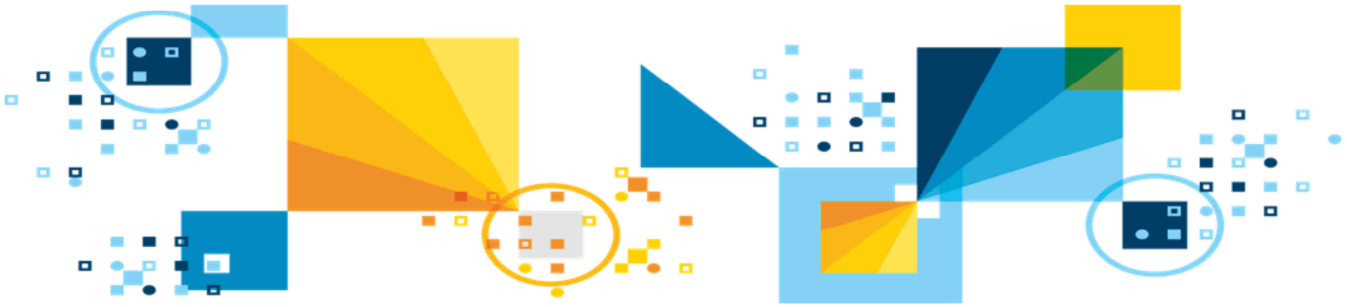
IBM Analytics

Paul Bird
pbird@ca.ibm.com

Db2 native encryption:
What to expect when you're expecting (to encrypt)



More and more customers are looking to leverage Db2 native encryption to help protect their database from evil-doers. Do you really understand what might change in your Db2 world as a result of moving to an encrypted database? To help avoid unpleasant surprises, this presentation reviews a list of things you need to consider before implementing database encryption

Agenda

- A brief overview of Db2 native encryption
- Performance considerations
- Availability considerations
- Operational considerations

What new availability requirements does an encrypted database bring to my environment?
How does encryption affect the performance of your system and what can you do about it?
How does an encrypted database affect your database operations?

A brief overview of Db2 native encryption

What is Db2 native encryption?

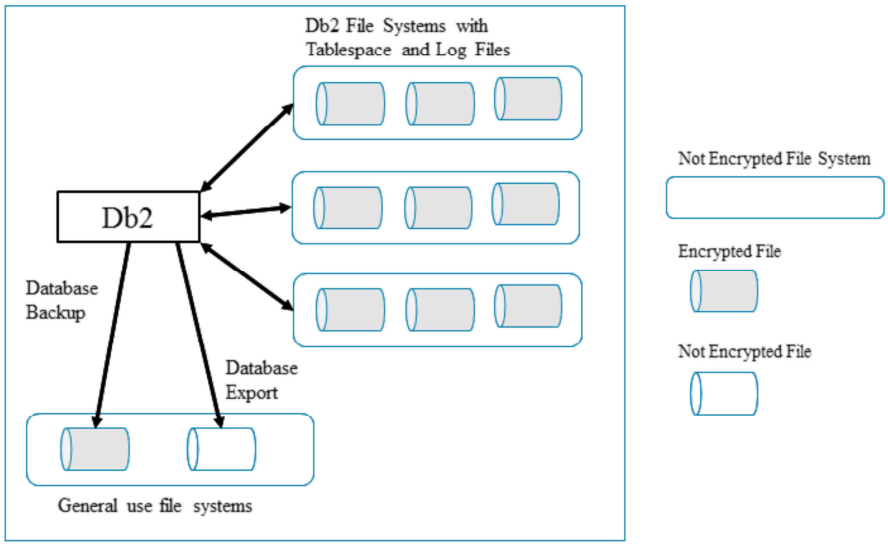
- **Db2 native encryption is an integrated approach to securing critical database files from outside access**
- **Db2 native encryption uses FIPS 140-2 certified encryption based on NIST SP 800-131A compliant cryptographic algorithms**
- **Db2 native encryption protects all critical database information:**
  - All table spaces (system defined and user defined)
  - Transaction logs (including logs in archives)
  - All backup images
  - LOAD COPY data
  - LOAD staging files

Db2 native encryption is about protecting your data from people outside the database whether it is in the database itself or in archived transaction logs or database backups.

Why would I be interested in Db2 native encryption?

- **I need to protect my database against threats to offline data such as:**
  - Users accessing data outside the scope of the database system (e.g. accessing the data files directly on the platform)
  - Theft or loss of physical media used to store database, backups, and logs (e.g. hard disks, USBs, tapes, etc.)
- **I need to protect my database and I need to protect that data no matter where it goes**
  - File system encryption protects the data in the file system: once data is moved from the file system, that data is no longer protected
- **I like the price!** ☺
  - Db2 native encryption is free on all Db2 11.1 offerings

Overview



This is a visual depiction of how Db2 native encryption protects the contents of specific files directly by encrypting the data written out and not the underlying file system. This means that when the file is moved, the data stays encrypted; there is no reliance on the file system itself being encrypted to protect the data.

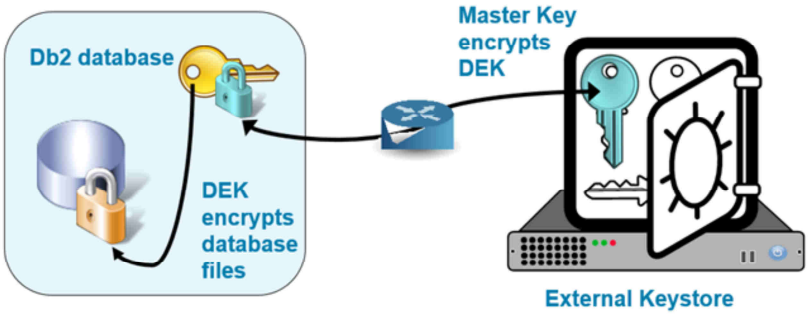
A couple of other key highlights

- **Simple to deploy**
- **Transparent to applications**
  - No application or schema changes required
- **All platforms and configurations of Db2 supported**
- **Automatically detects and uses CPU hardware acceleration when available**
  - Intel AES-NI hardware acceleration
  - Power 8 and Power 9 in-core support for the Advanced Encryption Standard (AES)
  - z – CPACF

Db2 native encryption is enabled with a new option on the create database command and applications and users are unaware of its existence. It works on all platforms supported by Db2 and in serial, DPF, and pureScale configurations. If there is CPU hardware acceleration available for encryption on your platform, Db2 native encryption will automatically detect and use it to dramatically reduce overhead.

How does Db2 native encryption work?

- **Db2 native encryption uses a 2-tier approach to data encryption**
  1. Db2 generates a unique Data Encryption Key (DEK) to secure the data
  2. The DEK is itself protected by encrypting it using an external Master Key (MK)
     - The encrypted DEK is kept with the data and is never exposed



Db2 native encryption uses an industry standard 2-tier approach to on-disk encryption. The master key is kept in an external keystore and Db2 accesses this keystore as needed.

The basic idea

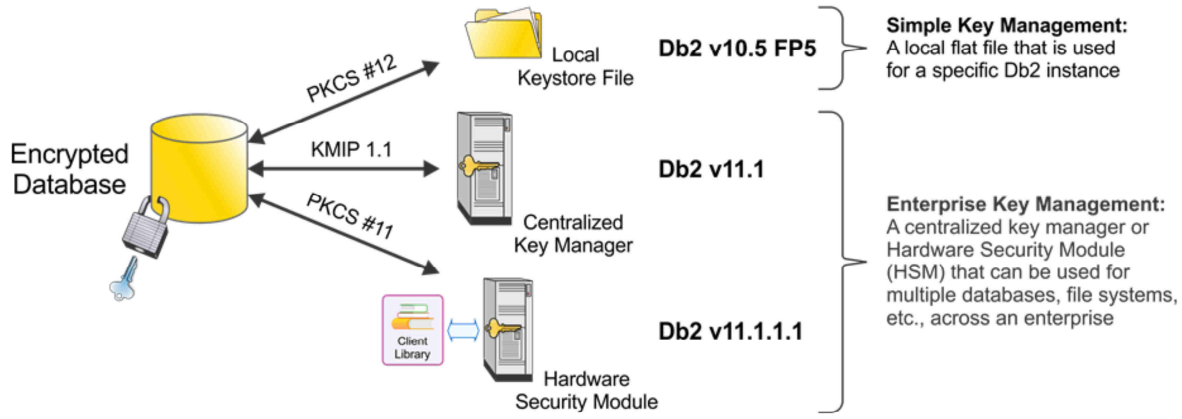
- Whenever secured data is written to disk by Db2, the DEK is used to encrypt the data
- Whenever secured data is read from disk by Db2, the DEK is used to decrypt the data
- Whenever access to the encrypted DEK is needed by Db2, the MK is accessed to decrypt the DEK

Warning: Acronyms ahead!

- **Organization for the Advancement of Structured Information Standards (OASIS)**
  - A global, non-profit consortium that works on open standards for security (among other things)
- **Public Key Cryptography Standards (PKCS)**
  - An OASIS standard for public key cryptography
  - The numbers 11 and 12 refer to specific parts of the standard
    - PKCS#12 is related to keystore files
    - PKCS#11 is related to access cryptographic tokens

- **Key Management Interoperability Protocol (KMIP)**
  - An OASIS standard for key management network protocol

External keystore options



The Master Key (MK) is managed externally and can be stored in a PKCS #12 compliant local keystore, which is created using the IBM Global Security Kit (GSKit). Db2 V11.1 added support for KMIP 1.1 compliant centralized key managers like IBM Security Key Lifecycle Manager (ISKLM) and Safenet KeySecure. And Db2 v11.1.1.1 offers direct support for Hardware Security Modules (HSMs) – integration with Gemalto (formerly Safenet) Luna SA HSM and Thales NShield Connect+ is provided.

Keystores supported

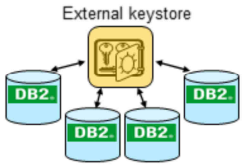
- **A PKCS#12 file**
- **Any keystore that supports KMIP 1.1 or higher**
- **These specific hardware security modules (HSM) which use PKCS#11:**
  - Gemalto Safenet HSM (formerly Luna) version 6.1 (firmware version 6.23.0) and above
  - Thales nShield HSM, security world software version 11.50 and above

Keystore in an HADR environment

- **All copies of the database must be encrypted and the same master key is used by both primary and standby**
  - They each have their own unique data encryption key
- **Both primary and standby require access to a keystore with same set of master keys**
  - Does not have to be the same keystore but can be (as long as the keystore itself is highly available)
- **If standby does not have access to the master key, *it will shut down***
- **Master key rotation is not logged and not transmitted independently to standby**
  - Detected via updated transaction header which means it happens when the next transaction is sent to the standby

Keystore in an multi-member environment (DPF, pureScale)

- **The same DEK and MK is used on all members in a DPF or pureScale deployment**
  - Each member requires access to a keystore for the master key
    - This may have licencing implications depending on the vendor
- **Ideally, you would use same keystore for all members to ensure consistency**
  - Need to ensure keystore is highly available to prevent single point of failure
  - If using a local keystore file (i.e. PKCS #12 file), can be a shared file or a copy



Deploying Db2 native encryption

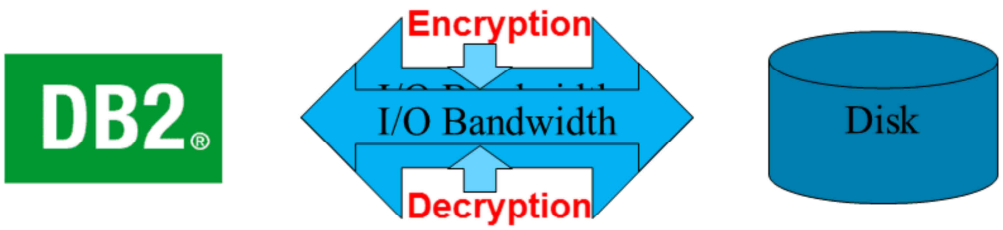
- **To encrypt an existing database, you effectively have to recreate and reload the database**
- **Possible approaches include:**
  - Restoring from a database backup
  - Loading from the old database into a new database
- **You can leverage an HADR relationship to reduce the outage time**
  - Standby is created as encrypted by restoring from a backup and then allowed to catch up to primary
  - Once caught up, standby can then take over from the primary allowing primary to be encrypted

HADR technote: https://www.ibm.com/developerworks/community/blogs/81c130c7-4408-4e01-adf5-658ae0ef5f0c/entry/Enable_DB2_native_encryption_in_an_HADR_environment?lang=en

Performance considerations

How does Db2 native encryption affect performance?

- **Encryption is applied each time Db2 reads/writes to disk and its overhead reduces the effective bandwidth of those actions**
  - Similar effect as replacing existing disk I/O system with a slower one



How much does encryption affect physical I/O?

- **The answer primarily depends on whether or not Db2 native encryption is able to leverage CPU hardware acceleration on your platform**
  - ➢ This is the most significant factor determining overall impact

- **Easiest way to find out if you have eligible hardware is to check your db2diag.log**
  - Detection is automatic
  - Intel-AES hardware has been leveraged since Db2 10.5 FP5 and Power8 support was introduced in Db2 10.5 FP9 and in Db2 11.1 GA.

db2diag.log entry (issued after db2start)

```
2018-10-03-22.29.29.779770-240 I40115A556          LEVEL: Event
PID     : 49677026             TID : 1             KTID : 126419151
PROC    : db2acd
INSTANCE: pbird                NODE : 000
HOSTNAME: hotelaix9
EDUID   : 1                    EDUNAME: db2acd
FUNCTION: DB2 Common, Cryptography, cryptContextRealInit, probe:1774
DATA #1 : String, 37 bytes
CPU flags(string): 0x0000000000000006
DATA #2 : String, 37 bytes
CPU flags(Uint64): 0x0000000000000006
DATA #3 : String, 40 bytes
PowerPC VCipher capability not available
DATA #4 : String, 1 bytes
```


The message in the third data element will tell you if CPU hardware acceleration is available for use by Db2 encryption. FYI, this also applies to SSL encryption.

How will this affect my workloads?

- **The answer depends on how sensitive your system is to increased physical I/O latency**
  - I/O intensive work will get hit the hardest (e.g. ETL, Backup, etc.)

- **A number of factors are involved:**
  - The amount and speed of CPU available for encryption/decryption
  - The amount of buffer pool page reuse by the workload(s)
  - The ability of the buffer pool background processes to keep up with the volume of physical read/writes
  - The amount of non-buffer pool I/O such as LOBs
  - The amount of parallelism involved in the physical read/write activities

How bad can it get?

- **Impact on I/O intensive work can be orders of magnitude (slower)**
  - Backup is particularly impacted (more on this later!)
- **Example from internal testing of backup/restore performance on an AIX Power7 system (no hardware acceleration):**
  - 5x degradation when using a single process
  - Improved to just under 30% when backup parallelism was used (e.g. a db2bm thread per tablespace)

Performance actions to consider

- **Plan to tune a newly encrypted system from scratch**
  - Starting assumption should be "encryption = slower physical I/O"
  - Look for new or exacerbated I/O bottlenecks in buffer pool interactions especially in I/O intensive work (e.g. ETL, backups, etc.)
- **Test all aspects of your workload and operations under native encryption**
  - Avoid surprises, look before you leap!
- **If additional CPU exists, consider increasing parallelism for any work doing the physical I/O where appropriate/possible**

If on AIX and still experiencing heavy overhead after enhancing I/O parallelism, consider adjusting AIX memory parameter to increase parallelism as outlined in this technote: http://www-01.ibm.com/support/docview.wss?uid=swg21983879

Important technote for AIX users on high concurrency systems

- "Malloc heap contention may cause performance degradation when using DB2 on AIX with specific features" (www.ibm.com/support/docview.wss?uid=ibm10741831)

Availability considerations

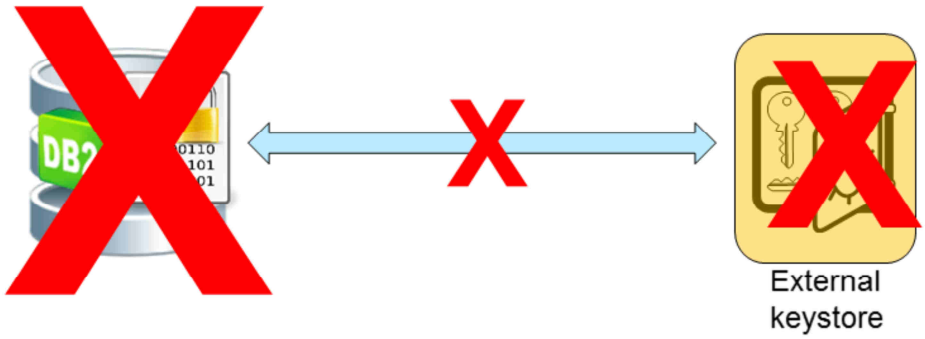
Words to live by in an encrypted world…

No master key = No access to data

- Your keystore is just as important as your database!

You need to plan how to keep your keystore available

- **While Db2 is running, it needs occasional access to the master key and if it can't get to the master key, you can't get to your data**
  - Keystore availability issues are data availability issues!



IBM Analytics

IBM

You need to think about how to recover your keystore

- **Db2 backups do not include the contents of the external keystore**
  - Responsibility for backing up and recovering the keystore and its contents falls upon ***you***
- **This also has implications for how a new Master Key is added and when it can be used**
  - After generating/inserting a new MK, you need to make sure there is a keystore backup containing the new value before you rotate the MK on the database!

➢ **Remember: No master key = No data!**

27

This is why the ALLOW_KEY_INSERT_WITHOUT_KEYSTORE_BACKUP configuration setting has a default value of false.

Keystore resiliency has to cover…

- **Keystore content (i.e. master keys)**
  - This is irreplaceable
- **Keystore access credentials (e.g. password)**
- **Keystore communication credentials (e.g. SSL certificate)**
- **Keystore client library (for HSM)**

External keystore

Operational considerations

Encryption brings additional operational responsibilities

- **Encryption adds to the scope of operations**
  - The protection and management of the external keystore is a net new addition to your operations
- **Encryption adds wrinkles to existing operations**
  - Need to securely handle keystore credentials
  - Need to adjust your DR strategy
- **Encryption may require changes to existing approaches to operations**
  - Disk compression techniques will no longer be effective
  - Backup/restore windows may no longer be adequate

Critical keystore planning aspects

- **Availability and recovery of the keystore**
  - Will be vendor and configuration specific
- **Network connections to keystore**
  - Including security, availability, latency
- **Capacity planning (e.g. licensing, performance)**
  - Will be vendor and configuration specific

Which keystore should we use?

- **The choice is yours and will be affected by:**
  - Corporate security strategy and budget
  - Existing standards/products in your organization
  - The outcome of vendor product evaluation for planning requirements

- **Although it appears to be the lowest cost option, we strongly recommend using a 3rd party product over a local PKCS #12 keystore file**
  - ➢ You will have to do everything for a local (PKCS #12) keystore file
  - – 3rd party products should have:
    - Flexible connection options (i.e. network based)
      - ➢ Useful for sharing keystore in HADR and multi-member Db2 configurations
    - Defined/integrated approaches for addressing backup and availability requirements

Handling keystore credentials

- **All keystores have their own access control mechanisms**
  - i.e. credentials such as a password must be presented to access the keystore
- **Since Db2 needs access to the keystore, it will also need access to the keystore credentials**
  - For local (PKCS #12) keystores, Db2 needs direct access to the keystore itself
  - For centralized keystores, Db2 needs access to the PKCS #12 keystore file containing the communication credentials for the keystore
- **The challenge is how to best make these credentials available to Db2 in your environment while keeping them secure?**

Db2 mechanisms for providing keystore credentials

- **"open keystore" option on db2start command**
  - "db2start open keystore using KeySt0rePassw0rd"
  - "db2start open keystore" ← will prompt for password
  - "db2start open keystore passarg <[FILENAME:<value> | FD:<value>]>"

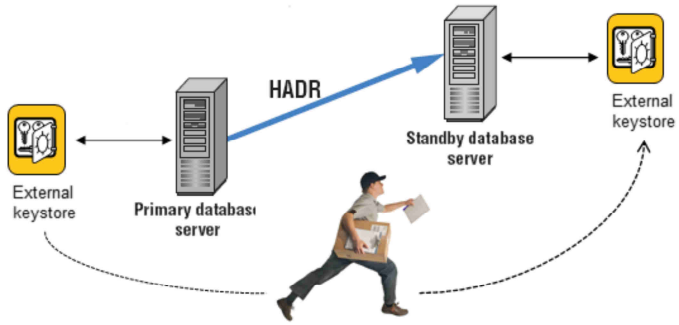
- **A "stash" file**
  - An automatic way to provide the password; Db2 will look for one if password not provided on db2start
  - Contains an obfuscated version of the keystore password
  - The stash file can only be read by the instance owner

How does encryption affect my DR strategy?

- **In order to access archived transaction log files and backup images, you need to have access to the same master key in use at the time that they were created**
  - ➢ You need to keep your keystore backups for as long as you keep your data
- **Recovery of a backup and roll forward through its related logs now requires that the appropriate master key(s) be in the keystore before recovery starts**
  - – New pre-requisite step for disaster recovery
- **Best practice: Never delete your master keys!**

Additional HADR considerations

- **If all members in an HADR relationship are not using a shared keystore, rotating the master key requires careful coordination**
  - New master key needs to be in standby keystore (and backed up!) before key rotation on the primary can begin



Encryption and compression

- **Encryption and compression do not get along**
  - The foundation of compression is finding repeating patterns in the data
  - The goal of good encryption is to effectively randomize the data

- **If you rely on disk de-duplication techniques to reduce storage consumption of backups and transaction logs, encryption will eliminate all your past gains**
- **To get the benefits of compression, it needs to happen before the encryption**
  - Either through active compression within the Db2 database or use of the combined compression and encryption backup library provided with Db2
- **Ember Crooks has a good blog entry on this topic:**
  - https://datageek.blog/2015/09/29/db2-backups-when-using-native-encryption/

Potential for increased backup/restore times

- **Backup and restore are very I/O intensive activities with limited caching effects**
  - Their performance will be hit hard by encryption
- **Each backup has its own unique data encryption key (i.e. not the same as database DEK)**
  - Backup and restore must pay the price to encrypt/decrypt between the different DEKs
- **The amount of parallelism available is limited (i.e. backup has one thread per tablespace)**
  - Limits the ability to leverage available CPU by spreading encryption overhead across multiple threads
- **Increased CPU consumption during backups as the result of using a software based approach to compression**

➢ **The end-result is that these operations will take longer, often significantly longer**

Summary

Summary of Db2 11.1 native encryption enhancements

- **Db2 11.1.0.0 GA**
  - Native encryption support for integration with KMIP 1.1 key management products
- **Db2 11.1.1.1**
  - Native encryption support for integration with Hardware Secure Modules (HSM) via PKCS #11
- **Db2 11.1.2.2**
  - Native encryption enhancement to support blacklisting “unavailable” keystores and other additional error handling configuration parameters (KMIP)
- **Db2 11.1.3.3**
  - Native encryption enhancement to significantly reduce keystore access for transaction logs
- **Db2 11.1.4.4**
  - Major rewrite of Db2 native encryption documentation

Db2 native encryption and you

- **Encryption of "data at rest" is becoming a fact-of-life for many of us**
  - The financial and reputational damage to your business from a data leak is huge!
- **Database encryption comes with an implicit cost for your organization in performance, availability, and operational considerations**
  - Planning ahead will make your life simpler and avoid unpleasant surprises
- **As long as you do your due diligence and proper preparation, encrypting your database does not have to be scary!**
- **Db2 native encryption provides a simple, comprehensive approach to encrypting your databases**
  - Give it a try!

Questions?