Encrypting your data through Db2 built-in functions 24

Last Updated: 2024-10-29

Db2 provides built-in data encryption and decryption functions that you can use to encrypt sensitive
24 such as credit card numbers and medical record numbers.

You can encrypt data at the column or value level. You must install the Integrated Cryptographic Service Facility (ICSF) to use the built-in functions for data encryption.

When you use most data encryption, Db2 requires the correct password to retrieve the data in a decrypted format. If an incorrect password is provided, Db2 does not decrypt the data. If the ENCRYPT_DATAKEY built-in function is used to encrypt data, you must have access to the key label stored with the

|< Db2 for z/OS offers two different sets of built-in functions for encrypting and decrypting data:

- Data encryption using the TDES algorithm: The ENCRYPT_TDES built-in function encrypts data using a password, and the DECRYPT_BIT, DECRYPT_CHAR and DECRYPT_DB built-in functions decrypt data using a password. The DECRYPT_BIT, DECRYPT_CHAR and DECRYPT_DB built-in functions decrypt data that was encrypted with the ENCRYPT_TDES built-in function. The user provides the password when invoking a decryption function
- Data encryption data using the 256-bit AES CBC algorithm: The ENCRYPT_DATAKEY built-in function encrypts data using the 256-bit AES CBC algorithm with either a random initialization vector (IV) or fixed initialization vector (IV) and a key label. The DECRYPT_DATAKEY_INTEGER, DECRYPT_DATAKEY_BIGINT, DECRYPT_DATAKEY_DECIMAL, DECRYPT_DATAKEY_VARCHAR, DECRYPT_DATAKEY_CLOB, DECRYPT_DATAKEY_VARGRAPHIC, DECRYPT_DATAKEY_DBCLOB, AND DECRYPT_DATAKEY_BIT built-in functions decrypt data that was encrypted using the ENCRYPT_DATA KEY built-in function. The decryption process uses the key label stored with the encrypted data to decrypt the data.

>|

Built-in encryption functions work for data that is stored within Db2 subsystem and is retrieved from within that same Db2 subsystem. The encryption functions do not work for data that is passed into and out of a Db2 subsystem. Application Transparent - Transport Layer Security (AT-TLS) is used to encrypt data between Db2 and applications or other data servers.

Attention: When the TDES encryption algorithm is used for encryption, Db2cannot decrypt data without the encryption password. If you forget the encryption password you cannot decrypt the data, and the data might become unusable.

- **Defining columns for data encrypted using the ENCRYPT_TDES built-in function.**
  When data is encrypted using the ENCRYPT_TDES built-in function, it is returned as a binary data

DB2 10.5 for Linux, UNIX, and Windows

DB2 native encryption

Last Updated: 2021-03-01

DB2® native encryption encrypts your DB2 database, requires no hardware, software, application, or schema changes, and provides transparent and secure key management. 24

Encryption is the process of transforming data into an unintelligible form in such a way that the original data either cannot be obtained or can be obtained only by using a decryption process. It is an effective way of protecting sensitive information that is stored on media or transmitted through untrusted communication channels. Encryption is mandatory for compliance with many government regulations and industry standards.

In an encryption scheme, the data requiring protection is transformed into an unreadable form by applying a cryptographic algorithm and an encryption key. A *cryptographic algorithm* is a mathematical function that is used in encryption and decryption processes. An *encryption key* is a sequence that controls the operation of a cryptographic algorithm and enables the reliable encryption and decryption of data.

Some data encryption solutions for protecting data at rest are suitable in cases of physical theft of disk devices, and some can protect against privileged user abuse. With *native database encryption*, the database system itself encrypts the data before it calls the underlying file system to write that data to disk. This means that not only your current data is protected, but also data in new table space containers or table spaces that you might add in the future. Native database encryption is suitable for protecting data in cases of either physical theft of disk devices or privileged user abuse.

A local or external key manager is typically used to manage the keys. A *database data encryption key* (DEK) is the encryption key with which actual user data is encrypted. A *master key* is a "key encrypting key": It is used to protect the DEK. Although the DEK is stored and managed by the database, the master key is stored and managed outside of the database.

These keys are shown in Figure 1, which provides an overview of DB2 native encryption.

Figure 1. An overview of DB2 native encryption