Cifrado de datos transparente (TDE)

Dirigido a: Administradores de Bases de Datos | Área: Bases de Datos
Autor: Pablo F. Dueñas | Servicios Profesionales Danysoft

Introducción

Una de las principales preocupaciones de los DBA es que puedan salir datos de la empresa. El típico ejemplo es que se guarden datos de tarjetas de crédito, cuentas bancarias o cualquier otra información personal. Incluso los datos de salarios o de precios podrían ser bastante problemáticos, y eso lo tienen todas las empresas. Esto es algo que se suele dejar al centrarse en la corrupción de datos, problemas de hardware, redes y cualquier cosa que pueda parar el servidor SQL. Sin embargo ahora es más relevante que nunca por la Ley de Protección de Datos.

A partir de SQL Server 2005, se implementaron cambios significativos para asegurarse que SQL Server es más seguro que las versiones anteriores. Entre los cambios se incluye una estrategia de «seguridad por diseño, seguridad de forma predeterminada y seguridad en la implementación» proyectada para ayudar a proteger la instancia del servidor y sus bases de datos de los ataques de seguridad. El cifrado transparente de datos se añade a la respuesta de Microsot en SQL Server 2008.

El cifrado de datos transparente

Hasta que salió SQL Server 2008 no había un método nativo de asegurar los ficheros físicos de la base de datos. Se podía usar el sistema operativo para asegurar directorios o ficheros o se podían asegurar celdas usando el cifrado de la versión 2005. También existía alguna herramienta para el cifrado de ficheros, pero eran adicionales y externos a SQL Server.

Podríamos definir el cifrado como el proceso de transformar información directamente inteligible, usando un algoritmo que lo hace ininteligible a cualquier otra persona que no tenga la clave. Hay varios métodos (algoritmos matemáticos) de cifrado, algunos mejores que otros. Aunque no son perfectos, en seguridad hay un principio muy importante: saber cuánto cuesta romper la clave y cuánto se obtendría de su ruptura. Es decir, si obtener los datos es tan costoso en términos de procesamiento, hardware, redes, etc., que no se va a obtener el beneficio suficiente, los datos están seguros.

Para definir correctamente el cifrado de datos transparente (TDE por su nombre en inglés: «Transparent Data Encryption») recurriremos a sus creadores, Microsoft: «*El cifrado de datos transparente (TDE) realiza el cifrado y descifrado de E/S en tiempo real de los datos y los archivos de registro. El cifrado utiliza una clave de cifrado de la base de datos (DEK), que está almacenada en el registro de arranque de la base de datos para que esté disponible durante la recuperación. La DEK es una clave simétrica protegida utilizando un certificado almacenado en la base de datos maestra del servidor o una clave asimétrica protegida por un módulo EKM. TDE protege los datos "en reposo", es decir, los archivos de datos y de registro».*

La principal ventaja respecto a otras tecnologías, es que no necesita cambios en la aplicación que se tenga desarrollada. Este permite a los DBA codificar la base de datos usando AES o 3DES sin tener que hacer ningún cambio a la aplicación cliente ni a los objetos de base de datos .

Cifrado de datos

Cuando se habilita el TDE, SQL Server inicia un hilo en segundo plano que examina todos los ficheros de la base de datos y los codifica. Este hilo crea un bloqueo compartido en la base de datos. El hilo de codificación se ejecuta asincrónicamente impidiendo únicamente la modificación de la estructura de archivos y poner la base de datos fuera de línea. Es decir, se efectúa de forma transparente mientras los usuarios acceden a la base de datos. La codificación también actúa sobre el fichero virtual de registro para que las siguientes escrituras al registro estén codificadas.

Como se ha dicho antes, los algoritmos de cifrado que se pueden usar son AES y Triple DES. En el primero se pueden usar claves de 128, 196 y 256 bits. Estos algoritmos no hacen que los ficheros de base de datos aumenten en tamaño, aunque sí que se usa relleno en el registro, por lo que estos ficheros sí serán más grandes.

Implementación

Comenzar a trabajar con el TDE es muy sencillo:

- Se crea una clave maestra.
- Se crea o se obtiene un certificado protegido por la clave maestra.
- Se crea una clave de base de datos que se protege con el certificado.
- Se habilita la base de datos para usar cifrado.

Veamos estos pasos uno a uno. Para crear la clave maestra, se ejecuta la instrucción CREATE MASTER KEY en la base de datos master. Si queremos ver si existen ya claves, tenemos a nuestra disposición la vista:

```
USE master;
GO

SELECT *
FROM sys.symmetric_keys;
GO
```

name	principal_id	symmetric_key_id	key_length	key_algorithm	algorithm_desc	c
##MS_ServiceMasterKey##	1	102	128	D3	TRIPLE_DES	2

Como vemos hay una clave por defecto para el servicio de SQL Server, pero no es la que buscamos, por lo que crearemos nuestra clave maestra:

Directiva de contraseñas

Antes de seguir, vamos a recordar cómo es una contraseña de SQL Server: puede contener hasta 128 caracteres, entre los que se permite incluir letras, símbolos y dígitos.
Como se supone que estas tareas las ejecute un DBA, no debería ser necesario insistir sobre el uso de contraseñas seguras. Sí vamos a recoger las recomendaciones de Microsoft:

- Debe tener una longitud de 8 caracteres como mínimo.
- Debe contener una combinación de letras, números y símbolos. En este punto añadir que no es conveniente sustituir la i con el 1, la o con el cero, etc.
- No es una palabra que pueda encontrarse en el diccionario. Esto incluye las recomendaciones siguientes: No es el nombre de un comando, una persona, un usuario o un equipo.
- Se cambia con frecuencia. Esta recomendación para las contraseñas de usuario, a veces no es posible para los certificados.
- Presenta diferencias notables con respecto a contraseñas anteriores.

Como sabemos, las directivas de complejidad de contraseñas se aplican a los inicios de sesión, pero también a todo tipo de contraseñas en SQL Server, y, en concreto a la creación de claves. Están diseñadas para impedir ataques por fuerza bruta mediante el aumento del número de contraseñas posibles. Según el sistema operativo en que se ejecuta SQL Server la directiva aplicada es más sencilla o más compleja.

Cuando se ejecuta SQL Server en Windows 2000, se impide la creación de contraseñas que sean:

- NULL o vacías.

- Iguales al nombre del equipo o el inicio de sesión.
- Una de las siguientes: «password», «admin», «administrator», «sa» o «sysadmin».

Por ejemplo, si queremos crear una clave usando una contraseña vacía nos sale el siguiente mensaje:

```
CREATE MASTER KEY ENCRYPTION
BY PASSWORD = '';
GO
```

Resultados | Mensajes

```
Mens. 15118, Nivel 16, Estado 1, Línea 2
Error de validación de contraseña. La contraseña no cumple los requisitos
de directiva de Windows porque no es bastante compleja.
```

Para sistemas operativos Windows 2003 y posteriores, las exigencias son más acordes con las recomendaciones expuestas anteriormente:

- No puede contener parte o todo el nombre de la cuenta del usuario. Una parte de un nombre de cuenta se define como tres o más caracteres alfanuméricos consecutivos delimitados en ambos extremos por un espacio en blanco –que incluye además del espacio, la tabulación, el retorno, etc.–, o por alguno de los siguientes caracteres: coma (,), punto (.), guión (-), carácter de subrayado (_) o signo de almohadilla (#).
- Debe tener una longitud de ocho caracteres como mínimo.
- Debe contener caracteres de tres de las siguientes categorías:
  - Letras en mayúsculas del alfabeto Latín (de la A a la Z).
  - Letras en minúsculas del alfabeto Latín (de la a a la z).
  - Dígitos en base 10 (del 0 al 9).
  - Caracteres que no sean alfanuméricos, como signo de admiración (!), signo de moneda ($), signo de almohadilla (#) o porcentaje (%). En caso de que se use una conexión ODBC, debe tenerse en cuenta que algunos caracteres no se pueden usar – en concreto [] {}() , ; ? * ! @– porque se utilizan para iniciar una conexión o para separar valores de conexión.

Clave maestra

Ahora ya nos podemos poner a crear la clave maestra:

```
USE master;
GO

CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'Pas$W0rd';
GO

SELECT *
FROM sys.symmetric_keys
WHERE name LIKE '%MS_DatabaseMasterKey%';
GO
```

name	principal_id	symmetric_key_id	key_length	key_algorithm	algorithm_desc
##MS_DatabaseMasterKey##	1	101	128	D3	TRIPLE_DES

Como podemos ver al volver a ejecutar la consulta anterior, ya existe una clave maestra de la base de datos. También vemos que el algoritmo que emplea es TRIPLE DES con una clave de 128 bits. Esto es porque es una clave maestra: otro tipo de claves permiten elegir el proveedor de codificación. También vemos que es

una clave simétrica: se usa para proteger las claves privadas de certificados y las claves asimétricas presentes en la base de datos. Es decir, va a proteger el certificado que creemos en el siguiente paso.
Para permitir el descifrado automático de la clave maestra, se cifra una copia de la clave mediante la clave maestra de servicio (la que hemos visto antes que crea automáticamente SQL Server ##MS_ServiceMasterKey## en la base de datos «master») y se almacena en la base de datos y en la base de datos maestra. En este caso, como la hemos creado en la base de datos maestra, sólo aparece en ella. Si lo hubiésemos hecho también en AdventureWorks, aparecería en ambas:

```
SELECT name, is_master_key_encrypted_by_server
FROM sys.databases;
```

name	is_master_key_encrypted_by_server
master	1
tempdb	0
model	0
msdb	0
ReportServer	0
ReportServerTempDB	0
AdventureWorks	1
AdventureWorksDW	0

La columna is_master_key_encrypted_by_server indica si la clave maestra de la base de datos se ha cifrado con la clave maestra de servicio.

Es importante realizar una copia de seguridad de la clave maestra utilizando BACKUP MASTER KEY y almacenar dicha copia en un lugar seguro.

Creación certificado

El segundo paso, como hemos indicado, es crear un certificado protegido por la clave maestra. Esto se hace de la manera siguiente:

```
CREATE CERTIFICATE CertificadoTDE
WITH SUBJECT = 'Certificado para TDE';
GO
```

Veamos los detalles del nuevo certificado:

```
SELECT *
FROM sys.certificates
WHERE [name] = 'CertificadoTDE';
GO
```

name	certificate_id	principal_id	pvt_key_encryption_type	pvt_key_encryption_type_desc
CertificadoTDE	258	1	MK	ENCRYPTED_BY_MASTER_KEY

is_active_for_begin_dialog	issuer_name
1	Certificado para TDE

subject	expiry_date	start_date
Certificado para TDE	2010-03-23 06:23:29.000	2009-03-23 06:23:29.000

Como estamos usando una clave simétrica para codificar el certificado, no usamos la opción ENCRYPTION BY PASSWORD. De esta manera se puede usar automáticamente. Como tampoco hemos especificado fechas, la de inicio es el momento en que se crea el certificado y la de expiración un año después. Como tampoco hemos especificado quién crea el certificado, usa el asunto del certificado.

Nada más señalar, que si se posee algún certificado externo, se pueden usar también con la misma instrucción, indicando dónde está. En caso de crearlo nosotros, se debe guardar en lugar seguro creando una copia de seguridad mediante BACKUP CERTIFICATE.

Clave de cifrado

Ahora que tenemos con qué proteger la clave de cifrado que se utiliza para cifrar de forma transparente una base de datos, podemos crearla:

```
USE AdventureWorks;
GO

CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE CertificadoTDE;
GO
```

Como en el paso anterior no hemos creado una copia de seguridad del certificado, nos da la advertencia: «*no se ha realizado una copia de seguridad del certificado usado para cifrar la clave de cifrado de la base de datos. Debería hacer inmediatamente una copia de seguridad del certificado y de la clave privada asociada con el certificado. Si el certificado llegara a no estar disponible o si tuviera que restaurar o asociar la base de datos en otro servidor, necesitará copias de seguridad tanto del certificado como de la clave privada. De lo contrario, no podrá abrir la base de datos*.»

Después de crear la copia de seguridad del certificado, podemos proseguir. Convendría señalar que antes se ha estado trabajando con la base de datos master, porque tanto la clave maestra como el certificado es obligatorio que estén en ella. Pero para cifrar una base de datos, la clave correspondiente debe estar en dicha base de datos, por eso hemos cambiado a AdventureWorks.

¿Por qué no hacemos copia de seguridad de la clave de cifrado? Porque no se puede exportar de la base de datos: Solo está disponible para el sistema, para los usuarios que tienen los permisos de depuración en el servidor y para los usuarios que tienen acceso a los certificados que cifran y descifran la clave de cifrado de la base de datos.

Para ver las características de la clave creada:

```
SELECT database_id, DB_NAME(database_id) AS nombre,
    encryption_state, key_algorithm, key_length
FROM sys.dm_database_encryption_keys;
```

database_id	nombre	encryption_state	key_algorithm	key_length
7	AdventureWorks	1	AES	128

Vemos que la clave pertenece a AdventureWorks, como hemos mencionado, que no está cifrada todavía (encryption_state = 1) y el algoritmo de codificación especificado.

Cifrado

Ya por fin podemos cifrar la base de datos. Tengamos en cuenta que ninguna de las etapas que nos exige SQL Server es gratuita. Es decir, la clave de cifrado la debe tener la base de datos para que el cifrado sea transparente. Por lo tanto, necesita un medio seguro para guardarla. Como tiene que ser una clave asimétrica o certificado que debe estar también en SQL Server para que siga siendo transparente, hay que guardar este certificado bajo llave. Para activar el cifrado usamos:

```
ALTER DATABASE AdventureWorks
SET ENCRYPTION ON;
GO
```

¿Cómo podemos saber si se está cifrando la base de datos? Ya hemos visto la consulta:

```
SELECT *
FROM sys.dm_database_encryption_keys;
```

database_id	encryption_state	key_algorithm	key_length	percent_complete
2	3	AES	256	0
7	2	AES	128	80,14706

¡Sorpresa!, no sólo cifra la base de datos AdventureWorks (cuyo identificador es 7 como sabemos), sino que también cifra la tempdb (identificador 2). Lógico, pues va a guardar los datos temporales de cualquiera de las bases de datos cifradas, y si no lo hiciera sería un agujero de seguridad. También vemos que la base de datos tempdb> ya está cifrada (estado = 3) y que AdventureWorks se está cifrando (estado = 2) y lleva ya un 80%. Cuando llegue al estado 3, habremos terminado.

Nota: el por ciento completado se refiere al cambio de estado. Una vez que ha terminado de cambiar el estado (de 2 a 3), es cero, porque no hay ningún cambio de estado en curso.

Verificación

Una forma fácil de verificar si se ha cifrado la base de datos es parar los servicios de SQL Server y abrir el fichero de datos. Si lo hacemos con una base de datos sin codificar, veremos que hay datos que se pueden leer a simple vista, aquellos con texto. Si la base de datos está cifrada veremos que todo es basura, imposible de leer, incluso el relleno de la base de datos.

También podemos probar hacer copia de seguridad y restaurar la base de datos. Primero hacemos la copia de seguridad:

```
BACKUP DATABASE AdventureWorks
TO DISK = N'Z:\Temp\AdventureWorks_Cifrada.bak'
WITH NOFORMAT, NOINIT,
    NAME = N'AdventureWorks completa cifrada',
    SKIP, NOREWIND, NOUNLOAD, STATS = 10;
GO
```

A continuación vamos a tratar de restaurar esta base de datos en otro servidor o instancia de SQL Server:

```
RESTORE DATABASE AdventureWorks
FROM DISK = N'Z:\Temp\AdventureWorks_Cifrada.bak'
WITH FILE = 1,
MOVE N'AdventureWorks' TO N'Z:\Temp\AdventureWorks.mdf',
MOVE N'AdventureWorks_log' TO N'Z:\Temp\AdventureWorks_log.ldf',
NOUNLOAD, STATS = 10;
GO
```

Esta instrucción, como esperábamos, falla con el siguiente mensaje:

```
Mens. 33111, Nivel 16, Estado 3, Línea 1
No se encuentra el servidor certificado con la
    huella digital '0xC7EA35FC0AF25F01FE9B8683224829A73556EC01'.
Mens. 3013, Nivel 16, Estado 1, Línea 1
Fin anómalo de RESTORE DATABASE.
```

De esta manera hemos podido ver que una copia de seguridad obtenida después del cifrado no se puede restaurar en otro servidor sin haber restaurado también el certificado.

¿Pero qué pasa si se obtienen los ficheros de la base de datos? Lo podemos comprobar copiando los ficheros de base de datos en frío e intentando adjuntarlos para leerlos:

```
CREATE DATABASE [AdventureWorks]
ON ( FILENAME = N'C:\Temp\AdventureWorks_Data.mdf'),
( FILENAME = N'C:\Temp\AdventureWorks_Log.ldf')
FOR ATTACH;
GO
```

Como era de esperar, nos da el mismo error que al restaurar:

```
Mens. 33111, Nivel 16, Estado 3, Línea 1
No se encuentra el servidor certificado con la
  huella digital '0xC7EA35FC0AF25F01FE9B8683224829A73556EC01'.
```

Restaurar una base de datos cifrada

Ya hemos visto que, una vez cifrada una base de datos, no hay acceso a los datos de forma clara ni a través de los ficheros de datos, de registro, datos temporales o copia de seguridad. Luego la única forma de restaurar o adjuntar la base de datos cifrada en otro servidor o instancia es añadiendo el mismo certificado. Para ello primero tenemos que hacer una copia de seguridad del certificado y de su clave privada:

```
USE master;
GO

BACKUP CERTIFICATE CertificadoTDE
TO FILE = 'C:\Temp\CertificadoTDE.cert'
WITH PRIVATE KEY
(
    FILE = 'C:\Temp\ClavePrivada.key',
    ENCRYPTION BY PASSWORD = 'P&a$Svvord'
);
GO
```

Ahora sí que ya podemos usar esta copia de seguridad del certificado para pasarlo a la nueva instancia o servidor. De ahí que se insista tanto sobre guardar en sitio seguro cualquier clave o certificado, pues de otra manera se tendría acceso público a los datos. Para la creación de la clave privada seguimos los pasos ya dichos de crear una clave maestra, certificado (en este caso lo importamos) y clave de cifrado (que también la importamos dentro del certificado). En la nueva instancia o servidor ejecutamos:

```
USE master;
GO

CREATE MASTER KEY ENCRYPTION
BY PASSWORD = 'Nv&va8a$s';
GO

CREATE CERTIFICATE NuevoCertificadoTDE
FROM FILE = 'C:\Temp\CertificadoTDE.cert'
WITH PRIVATE KEY
(
    FILE = 'C:\Temp\ClavePrivada.key',
    DECRYPTION BY PASSWORD = 'P&a$Svvord'
 );
GO
```

Como hemos creado el certificado y la clave privada en el nuevo servidor o instancia, ya se puede restaurar la base de datos cifrada:

```
RESTORE DATABASE NuevaAdventureWorks
FROM DISK = N'Z:\Temp\AdventureWorks_Cifrada.bak'
WITH FILE = 1,
MOVE N'AdventureWorks_data' TO N'C:\Temp\AdventureWorks_Data.mdf',
MOVE N'AdventureWorks_log' TO N'C:\Temp\AdventureWorks_Log.ldf',
NOUNLOAD, REPLACE, STATS = 10;
GO
```

Ahora sí que se restaura la base de datos con éxito, como podemos comprobar tanto por el mensaje de SQL Server como porque podemos navegar sin problemas por ella y ver sus datos. Por lo tanto, sólo se puede ver lo que contiene si se posee el certificado con la clave privada que protege los datos.

Qué tener en cuenta

Hay un problema conocido que afecta a TDE y a los grupos de archivos de sólo lectura. No se puede habilitar TDE cuando hay grupos de archivos de sólo lectura, lo cual es lógico, porque dejarían de ser archivos de sólo lectura. El estado de la base de datos se queda en «cifrado en progreso» y no termina. Por lo tanto, primero hay que permitir la escritura en todos los grupos de archivos, luego cifrar la base de datos y por último, volver a poner como sólo lectura los grupos de archivos correspondientes.

Respecto al tipo de datos FileStream, se puede cifrar la base de datos, pero los datos reales (los ficheros) en el sistema de archivos no se cifran. Para esto hay que usar algún sistema de cifrado de sistema operativo porque en realidad es él quien se encarga de estos ficheros o datos FileStream.

Por último, cualquier acción que implique restaurar la base de datos, moverla o hacer un espejo va a necesitar, obviamente, el certificado. Es, por tanto, muy importante que se guarden correctamente las copias de seguridad de los certificados, si no, las copias de seguridad de los datos no sirven ni se puede hacer un espejo.

Efectos secundarios

El beneficio fundamental de TDE es el cifrado nativo de todos los datos en el nivel físico. No se necesitan programas externos de cifrado, ni procedimientos almacenados o CLR, ni el sistema operativo: es nativo de SQL Server. Además protege no sólo ciertos datos, sino los ficheros completos.

Pero también tiene sus «problemas». El primero ya lo hemos visto: se cifra la base de datos tempdb, que es usada por todas las bases de datos de la instancia. Aunque el cifrado no supone mucha degradación de la respuesta del servidor, cada vez que una base de datos abierta (no cifrada) use datos temporales, van a tener que ser cifrados y descifrados, lo que afecta a su rendimiento.

Otro «problema» son las copias de seguridad comprimidas. Una base de datos cifrada comprime muy poco. Esto es de esperar si aceptamos que lo que hay en los ficheros debe ser completamente aleatorio, sin patrones que indiquen qué hay detrás y, por tanto, sin patrones comprimibles. Además, no sólo no va a haber prácticamente diferencia en tamaño entre una copia de seguridad comprimida y sin comprimir, sino que también va a haber porca diferencia en tiempo.

Por último, una advertencia que se ha lanzado en los foros es que, una vez que se cifra una base de datos, aunque luego se quite el cifrado, siempre hay una referencia al certificado y se va a necesitar en la instancia donde se restaure. De otro modo dará error. Por si acaso la versión que se usa tiene este problema, mejor comprobarlo antes con un test, y tenerlo presente para no perder la copia del certificado.

Conclusión

Hemos visto que el cifrado transparente de datos trabaja bien y es muy fácil de instalar. Nos da cifrado nativo completamente transparente para el usuario, de todos los datos independiente del sistema operativo o servicios de terceros. Cifra todos los datos sin preocuparnos si cambiamos de sitio o creamos

algún fichero. De esta manera se pueden cumplir con los requisitos de privacidad del estado o internos de la compañía, con claves y certificados fuertes, sin aumentar el tamaño de la base de datos.

Pero no debemos olvidar que tiene una llave sin la cual los datos se pueden convertir en basura. Por tanto, lo mejor es tener varias copias del certificado almacenadas en lugares seguros: no sólo porque no lo puedan robar o no se pueda perder en un incendio, sino también porque el medio donde está almacenado no se degrade y se vuelva ilegible. De esta manera, si tenemos varias copias repartidas, en caso de perder alguna por la causa que sea, siempre dispondremos de otra.

Por último un pensamiento: Esto es muy útil para bases de datos portátiles, pero sólo viene en la edición Enterprise.

Para más información.

Danysoft, es Gold certified Partner de Microsoft ofreciéndole tanto las licencias de SQL Server en las mejores condiciones como los servicios de formación y consultoría necesarios para su correcto uso. Puede contactar con Danysoft en el 902 123146, o ver más información en www.danysoft.com