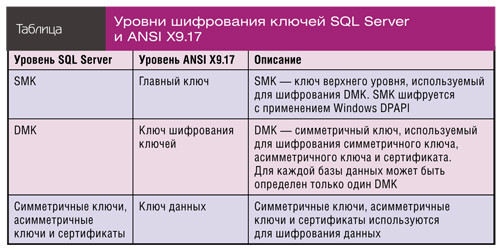
Шифрование в базах данных SQL Server

Законы требуют шифрования конфиденциальной информации на уровне базы данных и операционной системы. SQL Server, как и другие распространенные коммерческие системы управления базами данных, располагает множеством вариантов шифрования, в том числе на уровне ячеек, базы данных и файлов через Windows, а также на транспортном уровне. Эти варианты шифрования обеспечивают безопасность информации на уровне базы данных и операционной системы. Кроме того, они снижают вероятность несанкционированного раскрытия конфиденциальных сведений, даже если поражены инфраструктура или база данных SQL Server.

### Модель шифрования SQL Server

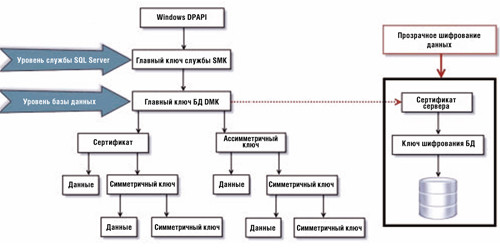
Модель шифрования SQL Server в основном предоставляет функции управления ключами шифрования, соответствующие стандарту ANSI X9.17. В этом стандарте определены несколько уровней ключей шифрования, использующихся для шифрования других ключей, которые в свою очередь применяются для шифрования собственно данных. В таблице перечислены уровни ключей шифрования SQL Server и ANSI X9.17.



Главный ключ службы Service master key(SMK) — ключ верхнего уровня и предок всех ключей в SQL Server. SMK — асимметричный ключ, шифруемый с использованием Windows Data Protection API (DPAPI). SMK автоматически создается, когда шифруется какой-нибудь объект, и привязан к учетной записи службы SQL Server. SMK используется для шифрования главного ключа базы данных Database master key (DMK).

Второй уровень иерархии ключей шифрования — DMK. С его помощью шифруются симметричные ключи, асимметричные ключи и сертификаты. Каждая база данных располагает лишь одним DMK.

Следующий уровень содержит симметричные ключи, асимметричные ключи и сертификаты. Симметричные ключи — основное средство шифрования в базе данных. Microsoft рекомендует шифровать данные только с помощью симметричных ключей. Кроме того, в SQL Server 2008 и более новых версиях есть сертификаты уровня сервера и ключи шифрования базы данных для прозрачного шифрования данных. На рисунке 1 показана иерархия ключей шифрования



Начиная с SQL Server 2005, можно шифровать или расшифровывать данные на сервере. Делать это можно различными способами. Например, можно шифровать данные в базах данных одним из следующих методов.

* Пароль. Это наименее надежный способ, так как для шифрования и расшифровки данных используется одна и та же парольная фраза. Если хранимые процедуры и функции не зашифрованы, то доступ к парольной фразе возможен через метаданные.
* Сертификат. Этот способ обеспечивает надежную защиту и высокое быстродействие. Сертификат можно связать с пользователем; подписать его необходимо с помощью DMK.
* Симметричный ключ. Достаточно надежен, удовлетворяет большинству требований к безопасности данных и обеспечивает достаточное быстродействие. Для шифрования и расшифровки данных используется один ключ.
* Асимметричный ключ. Обеспечивает надежную защиту, так как применяются различные ключи для шифрования и расшифровки данных. Однако это негативно влияет на быстродействие. Специалисты Microsoft не рекомендуют использовать его для шифрования крупных значений. Асимметричный ключ может быть подписан с использованием DMK или создан с помощью пароля.

SQL Server располагает встроенными функциями для шифрования и расшифровки на уровне ячеек. Функции шифрования:

ENCRYPTBYKEY, использует симметричный ключ для шифрования данных;

ENCRYPTBYCERT, использует открытый ключ сертификата для шифрования данных;

ENCRYPTBYPASSPHRASE, использует парольную фразу для шифрования данных;

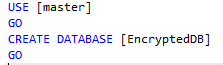
ENCRYPTBYASYMKEY, использует асимметричный ключ для шифрования данных.

Функции расшифровки:

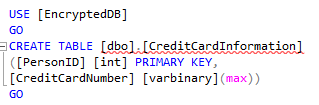
* DECRYPTBYKEY, использует симметричный ключ для расшифровки данных;
* DECRYPTBYCERT, использует открытый ключ сертификата для расшифровки данных;
* DECRYPTBYPASSPHRASE, использует парольную фразу для расшифровки данных;
* DECRYPTBYASYMKEY, использует асимметричный ключ для расшифровки данных;
* DECRYPTBYKEYAUTOASYMKEY, использует асимметричный ключ, который автоматически расшифровывает сертификат.

SQL Server располагает двумя системными представлениями, с помощью которых можно получить метаданные для всех симметричных и асимметричных ключей, существующих в экземпляре SQL Server. Как видно из названий, sys.symmetric\_keys возвращает метаданные для симметричных, а sys.asymmetric\_keys — для асимметричных ключей. Еще одно полезное представление — sys.openkeys. В этом представлении каталога содержится информация о ключах шифрования, открытых в текущем сеансе.

В первую очередь создайте базу данных EncryptedDB с помощью среды SQL Server Management Studio (SSMS) или выполнив программный код T-SQL:

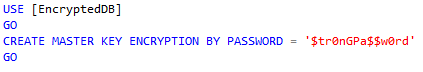


Затем запустите код T-SQL для создания таблицы с именем CreditCardInformation в базе данных EncryptedDB:

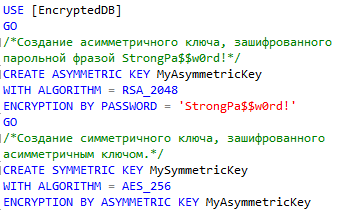


Эта таблица будет содержать ложную информацию о кредитных картах. Номера кредитных карт будут сохранены в столбце двоичных переменных, потому что они будут шифроваться.

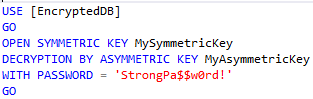
Затем используйте следующий программный код для создания главного ключа DMK базы данных EncryptedDB, шифруемого с помощью парольной фразы $tr0nGPa$$w0rd:



Данные шифруются с использованием симметричного ключа, который будет зашифрован с помощью асимметричного ключа. Для этого необходимо создать асимметричный ключ, зашифровать его парольной фразой $tr0nGPa$$w0rd, создать симметричный ключ и зашифровать симметричный ключ с помощью только что созданного асимметричного ключа.



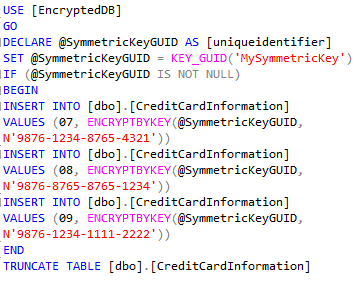
Теперь мы можем приступить к шифрованию данных. Для этого необходимо сначала открыть симметричный ключ, только что созданный с помощью команды OPEN SYMMETRIC KEY, за которой следует имя симметричного ключа. Затем вы указываете, что нужно расшифровать его с использованием заданного асимметричного ключа. Программный код выглядит следующим образом:



После выполнения этого кода направьте запрос в представление sys.openkeys, чтобы убедиться, что ключ открыт:



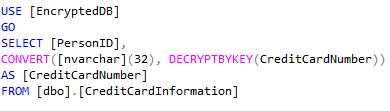
Необходимо ввести несколько номеров кредитных карт в таблицу CreditCardInformation, запустив код:

****

Затем направьте запрос к таблице CreditCardInformation:

****

Все данные в столбце CreditCardNumber представлены в двоичном формате. С помощью функции DECRYPTBYKEY можно просмотреть зашифрованные данные:

****

### Преимущества и недостатки шифрования на уровне ячеек

У шифрования на уровне ячеек есть свои достоинства и недостатки. Среди достоинств — более детальный уровень шифрования, что позволяет зашифровать единственную ячейку внутри таблицы. Кроме того, данные не расшифровываются, пока не придет время их использовать, то есть данные из загруженной в память страницы зашифрованы. Можно назначить ключ пользователям и защитить его паролем, чтобы предотвратить автоматическую расшифровку.

Среди недостатков шифрования на уровне ячеек — необходимость изменения схемы, так как все зашифрованные данные должны быть сохранены с использованием типа данных varbinary. Кроме того, снижается общая производительность базы данных из-за дополнительной обработки при шифровании и расшифровке данных. Требует времени и просмотр таблицы, поскольку индексы для таблицы зашифрованы и не могут быть использованы.

Прозрачное шифрование данных

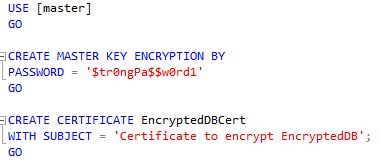
В SQL Server появилась возможность зашифровать всю базу данных с использованием прозрачного шифрования. При таком шифровании можно защитить базы данных без изменения существующих приложений, структур баз данных или процессов. Это лучший вариант для выполнения требований нормативных актов и правил корпоративной безопасности, поскольку шифруется вся база данных на жестком диске.

Прозрачное шифрование данных шифрует базы данных в реальном времени, по мере внесения записей в файлы (\*.mdf) базы данных SQL Server и файлы (\*.ldf) журнала транзакций. Записи также шифруются в реальном времени во время резервного копирования базы данных, а затем формируются моментальные снимки. Данные шифруются перед записью на диск и расшифровываются перед извлечением. Процесс полностью прозрачен для пользователя или приложения, поскольку выполняется на уровне SQL Server Service.

При прозрачном методе SQL Server шифрует базу данных с помощью ключа шифрования базы данных. Этот асимметричный ключ хранится в загрузочной записи базы данных и потому всегда доступен при восстановлении.

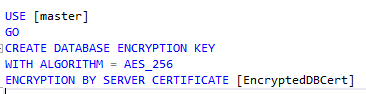
Подготовка к применению прозрачного шифрования данных

Чтобы разрешить прозрачное шифрование данных, необходимо создать DMK и сертификат сервера в базе данных master. Для этого выполните следующий программный код:

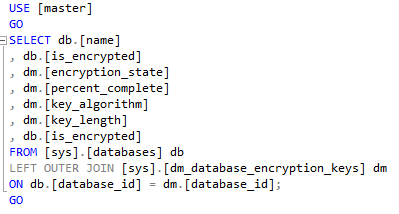


Важно немедленно сделать резервную копию сертификата и ключа DMK, связанного с сертификатом. Если сертификат становится недоступным или нужно восстановить или присоединить базу данных к другому серверу, то потребуются резервные копии как сертификата, так и DMK. В противном случае открыть базу данных не удастся.

Последний шаг — создать ключ шифрования базы данных и включить шифрование для базы данных, которую нужно защитить.



Чтобы узнать состояние шифрование всех баз данных на сервере, выполните запрос:



В этом запросе используется представление динамического управления с именем sys.dm\_database\_encryption\_keys, чтобы выяснить состояние шифрования каждой базы данных. Как показано на рисунке 5, результат содержит информацию о ключе шифрования базы данных для каждой базы данных.

Шифрование на транспортном уровне

В SQL Server предусмотрено два варианта шифрования данных, которые будут передаваться по сети между экземпляром SQL Server и клиентским приложением.

* Ipsec. Реализован на уровне операционной системы и обеспечивает проверку подлинности с использованием Kerberos, сертификатов и общих ключей. IPsec обеспечивает прозрачные для приложений службы шифрования с надежной фильтрацией для блокирования трафика по протоколам и портам. IPsec можно настроить с помощью локальной политики безопасности или групповой политики. IPsec может применяться с Windows 2000 и более новыми версиями. Выбирая этот метод, необходимо убедиться, что операционные системы как клиентов, так и сервера совместимы с протоколом IPsec.
* SSL. SSL настраивается на SQL Server. Он наиболее широко применяется для поддержки веб-клиентов, но может использоваться и для собственных клиентов SQL Server. SSL проверяет сервер, когда клиент запрашивает шифрованное соединение. Если экземпляр SQL Server функционирует на компьютере с сертификатом от публичного удостоверяющего центра, то удостоверение компьютера и экземпляр SQL Server гарантируют, что цепочка сертификатов ведет к корневому центру сертификации. Для такой проверки на стороне сервера требуется, чтобы компьютер, на котором функционирует клиентское приложение, доверял корневому удостоверяющему центру, используемому сервером. Возможно шифрование с использованием самозаверяющего сертификата, но защита самозаверяющего сертификата ненадежна.

Защитим данные в SQL Server

Шифрование — процесс кодирования конфиденциальных данных с использованием ключа или пароля. Шифрование надежно защищает данные и сокращает вероятность несанкционированного раскрытия конфиденциальной информации, так как без соответствующего ключа или пароля данные бесполезны. SQL Server располагает многими режимами шифрования, в том числе на уровне ячеек, базы данных, файлов через Windows и шифрования на транспортном уровне. Шифрование SQL Server не решает проблему доступности инфраструктуры и баз данных SQL Server, но повышает защищенность данных на уровнях базы данных и операционной системы, даже если нарушена конфиденциальность инфраструктуры или баз данных SQL Server.