МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ЛЕКЦИОННЫЕ МАТЕРИАЛЫ (ПРЕЗЕНТАЦИИ К ЛЕКЦИОННЫМ МАТЕРИАЛАМ)

Безопасность систем баз данных

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень	специалист *(бакалавриат, магистратура, специалитет)*
Форма обучения	очная *(очная, очно-заочная, заочная)*
Направление(-я) подготовки	10.03.01 «Информационная безопасность автоматизированных систем» *(код и наименование)*
Институт	Кибербезопасности и цифровых технологий *(полное и краткое наименование)*
Кафедра	Информационно-аналитические системы кибербезопасности (КБ-2) *(полное и краткое наименование кафедры, реализующей дисциплину (модуль))*
Лектор	К.т.н., доцент Шукенбаев Айрат Бисенгалеевич *(сокращенно – ученая степень, ученое звание; полностью – ФИО)*

Используются в данной редакции с учебного года 2023/2024
(учебный год цифрами)

Проверено и согласовано «____» _____20___г.

_____ А.А. Бакаев
(подпись директора Института/Филиала с расшифровкой)

Москва 2024 г.

Ощущение полной безопасности наиболее опасно.
Илья Нисонович Шевелев
Везде, где есть жизнь, есть и опасность.
Ральф Уолдо Эмерсон

Безопасность систем баз данных.

Тема лекции: Шифрование в базах данных

Подходы к шифрованию баз данных. Прозрачное/внешнее шифрование базы данных. Шифрование на уровне столбца. Шифрование на уровне поля. Шифрование на уровне файловой системы. Полное шифрование диска. Симметричное и асимметричное шифрование базы данных. Ключевой менеджмент. Хеширование. Соление. Перец. Шифрование на уровне приложений. Риски шифрования базы данных. Шифрование в базах данных MS SQL Server.

Подходы к шифрованию баз данных

Акт Грэмма-Лича-Блили (GLBA). *Требования GLBA* ***для защиты конфиденциальных данных*** *потребителей применяются к финансовым учреждениям, которые предлагают клиентам финансовые продукты и услуги, такие как кредиты, консультации по инвестициям и страховое обслуживание.*

Решением Европейского Парламента и Совета был принят новый закон о защите персональных данных - Общий/Генеральный ***регламент по защите персональных данных (GDPR - General Data Protection Regulation****). По сути он представляет собой два документа:*

- Регламент (EU) 2016/679 Европейского Парламента и Совета «О защите физических лиц в отношении обработки персональных данных и о свободном перемещении таких данных и отмене Директивы 95/46/ ЕС (Общие правила защиты данных)»
- *Директиву (EU) 2016/680 Европейского Парламента и Совета «О защите физических лиц в отношении обработки персональных данных компетентными органами в целях предотвращения, расследования уголовных преступлений, ведения разыскных или судебных действий или исполнения уголовных наказаний, а также за свободное перемещение таких данных и отменяя Рамочное решение Совета 2008/977 / JHA»*

Акт о преемственности и подотчетности медицинского страхования (HIPAA) *- это профайл защиты, направленный на защиту электронно-защищенной информации о состоянии здоровья* ***(EPHI)****.*

Payment Card Industry Data Security Standard (PCI DSS) .

Закон Sarbanes-Oxley (SOX).

Шифрование базы данных – использование технологии шифрования для преобразования информации, хранящейся в базе данных (БД), в шифротекст, что делает ее прочтение невозможным для лиц, не обладающих ключами шифрования.

Для шифрования базы данных доступно несколько методов и технологий.

Подход к шифрованию может существенно отличаться в рамках конкретных СУБД. Однако все применяемые методы основаны на широко известных общих подходах, принцепах и стандартах шифрования данных.

Прозрачное/внешнее шифрование базы данных.

*Прозрачное шифрование данных (transparent data encryption - **TDE**) — технология по шифрованию баз данных, которая обеспечивает безопасность, основанную на стандартах, защищает данные в сети, на диске и на носителе резервного копирования, и может использоваться для обеспечения высокого уровня безопасности для столбцов, таблиц и табличных пространств, которые представляют собой файлы базы данных, хранящиеся на жёстких дисках или гибких дисках или компакт-дисках, и другую информацию, требующую защиты.*

Шифрование на уровне столбца.

Типичная реляционная база данных разделена на таблицы, разделенные на столбцы, в каждой из которых есть строки данных. Хотя TDE обычно шифрует всю базу данных, шифрование на уровне столбцов позволяет шифровать отдельные столбцы в базе данных.

Во-первых, возможность шифрования отдельных столбцов позволяет шифрованию на уровне столбца быть значительно более гибким по сравнению с системами шифрования, которые шифруют всю базу данных.

Во-вторых, можно использовать полностью уникальный и отдельный ключ шифрования для каждого столбца в базе данных.

Недостаток, связанный с шифрованием базы данных на уровне столбцов - скорость или ее потеря.

Шифрование на уровне поля.

Можно шифровать всю базу данных, а можно шифровать содержимое отдельных полей в таблицах. Шифровать следует только символьные поля или поля типа MEMO. Проводятся экспериментальные работы по обеспечению операций базы данных (таких как поиск или арифметические операции) над зашифрованными полями без необходимости их расшифровки. Для рандомизации требуется надежное шифрование - каждый раз должен генерироваться другой результат. Это известно, как вероятностное шифрование. Шифрование на уровне поля позволяет пользователям проверять равенство без дешифрования данных.

Недостатки метода. На это уходит время. Нельзя напрямую вводить в таблицы и формы данные, подлежащие шифрованию.

Шифрование на уровне файловой системы.

Шифрование на уровне файловой системы (Filesystem-level encryption) называемое шифрованием на основе файлов, FBE или шифрованием файлов/папок, представляет собой форму шифрования диска, при которой отдельные файлы или каталоги шифруются самой файловой системой. Это отличается от полного шифрования диска, когда зашифровывается весь раздел или диск, на котором находится файловая система. Типы шифрования на уровне файловой системы включают: использование «наращиваемой» криптографической файловой системы, расположенной поверх основной файловой системы единая файловая система общего назначения с шифрованием. *К преимуществам* шифрования на уровне файловой системы относятся: гибкое управление ключами на основе файлов , так что каждый файл обычно может быть зашифрован отдельным ключом шифрования индивидуальное управление зашифрованными файлами.

*Технология **FBE** основана на технологии шифрования на уровне файлов. По сравнению с технологией FDE, основанной на шифровании на уровне диска, производительность системы смартфона выше, а шифрование данных более гибкое.*

Файловые системы общего назначения с шифрованием - включают шифрование на уровне файловой системы и обычно не шифруют метаданные файловой системы, такие как структура каталогов, имена файлов, размеры или временные метки модификации

***ZFS** — это современная файловая система, которая использует собственный механизм кэширования чтения, а не полагается на кэш страниц операционной системы для хранения копий недавно прочитанных блоков в оперативной памяти.*

***Access Control List или ACL** — список управления доступом, который определяет, кто или что может получать доступ к объекту (программе, процессу или файлу), и какие именно операции разрешено или запрещено выполнять субъекту (пользователю, группе пользователей).*

***CryFS** – это криптографическая файловая система. Бесплатный инструмент шифрования с открытым исходным кодом, созданный специально для облачного хранилища.*

***EncFS** — свободная криптографическая файловая система, основанная на FUSE, прозрачно шифрующая файлы, используя произвольный каталог в качестве места для хранения зашифрованных файлов. Распространяется под лицензией GPL.*

Криптографические файловые системы - это специализированные (не универсальные) файловые системы, специально разработанные с учетом требований шифрования и безопасности. Обычно они шифруют все содержащиеся в них данные, включая метаданные.

Шифрованная файловая система (EFS) – технология, которая позволяет прозрачно шифровать файлы для защиты конфиденциальных данных от злоумышленников, имеющих физический доступ к компьютеру. *EFS* доступна во всех версиях *Windows*, кроме домашних, начиная с Windows 2000. По умолчанию файлы не шифруются, но пользователи могут включить шифрование для каждого файла, каталога или диска. Некоторые параметры *EFS* также могут быть заданы с помощью групповой политики в доменных средах Windows

Полное шифрование диска (Full Disk Encryption -FDE) - это технология, которая защищает информацию, преобразуя ее в нечитаемый код, который не могут легко расшифровать неуполномоченные лица.

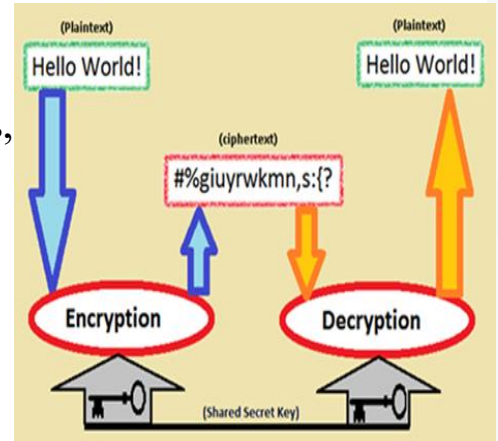
В Windows 10, 8.1, 8 и 7 есть *BitLocker* для полнодискового шифрования. *BitLocker Drive Encryption* — проприетарная технология шифрования дисков, являющаяся частью операционных систем.

Симметричное шифрование базы данных.

Симметричное шифрование в контексте шифрования базы данных включает закрытый ключ, применяемый к данным, которые хранятся и вызываются из базы данных. Этот закрытый ключ изменяет данные таким образом, что делает их нечитаемыми без предварительной расшифровки.

Недостатком симметричного шифрования является возможность утечки конфиденциальных данных, если закрытый ключ будет распространен среди лиц, которые не должны иметь доступа к данным.

Преимущество. Учитывая, что в процессе шифрования задействован только один ключ, в целом можно сказать, что скорость является преимуществом симметричного шифрования.



Асимметричное шифрование базы данных.

Асимметричное шифрование расширяет симметричное шифрование за счет включения в метод шифрования двух разных типов ключей: закрытых и открытых ключей. Открытый ключ может быть доступен любым и является уникальной для одного пользователя, в то время как секретный ключ является секретным ключом, который является уникальным и известен одному пользователю. В большинстве сценариев открытый ключ является ключом шифрования, тогда как закрытый ключ является ключом дешифрования.

Ключевой менеджмент.

Управлением ключами. Если ключи шифрования не управляются и не хранятся должным образом, может произойти утечка очень конфиденциальных данных. Кроме того, если система управления ключами удаляет или теряет ключ, информация, которая была зашифрована с помощью указанного ключа, по существу также оказывается «потерянной».

Хеширование.

Хеширование используется в системах баз данных как метод защиты конфиденциальных данных, таких как пароли; однако он также используется для повышения эффективности обращения к базе данных. Введенные данные обрабатываются алгоритмом хеширования. Алгоритм хеширования преобразует введенные данные в строку фиксированной длины, которую затем можно сохранить в базе данных. У систем хеширования есть две критически важные характеристики:

Во-первых, хэши «уникальны и повторяемы».

Во-вторых, алгоритмы хеширования необратимы.

Соление - это добавление случайных данных перед их передачей через хеш-функцию, и они чаще всего используются с паролями.

Недостатки соли. Соление теряет свою эффективность, если оно сделано неправильно. Две наиболее распространенные проблемы возникают, когда соли слишком короткие, или если они не уникальны для каждого пароля. Короткие соли по-прежнему уязвимы для атак радужного стола, потому что они не делают получающийся хэш достаточно редким. Если соли используются повторно для каждого хешированного пароля, и соль обнаруживается, это значительно упрощает определение каждого пароля в базе данных. Использование

Шифрование на уровне приложений

При шифровании на уровне приложений процесс шифрования данных завершается приложением, которое использовалось для создания или изменения данных, которые должны быть зашифрованы. По сути, это означает, что данные шифруются перед записью в базу данных

Преимущества. Первое - шифрование на уровне приложений может упростить процесс шифрования, используемый компанией. Если приложение шифрует данные, которые оно записывает/изменяет из базы данных, то дополнительный инструмент шифрования не нужно будет интегрировать в систему. Второе - касается общей темы воровства. Учитывая, что данные шифруются перед записью на сервер, хакеру потребуется доступ к содержимому базы данных, а также к приложениям, которые использовались для шифрования и дешифрования содержимого базы данных, чтобы расшифровать конфиденциальные данные.

Недостатки.

1. приложения, используемые фирмой, необходимо будет модифицировать для шифрования самих данных. Это может отнять значительное количество времени и других ресурсов. Учитывая характер альтернативных издержек, компании могут не поверить, что шифрование на уровне приложений стоит вложенных средств.
2. шифрование на уровне приложения может ограничивать производительность базы данных. Если все данные в базе данных зашифрованы множеством различных приложений, становится невозможным индексирование или поиск данных в базе данных.
3. сложность управления ключами возрастает, поскольку несколько разных приложений должны иметь полномочия и доступ для шифрования данных и записи их в базу данных.

Шифрование в базах данных MS SQL Server.

- модель шифрования SQL Server, включая иерархию шифрования SQL Server и концепции серверных сертификатов и ключей шифрования баз данных;
- прозрачное шифрование данных, позволяющее незаметно шифровать всю базу данных сразу, не затрагивая интерфейсных приложений и приложений среднего слоя;
- расширяемое управление ключами, позволяющее использовать аппаратные модули защиты от независимых поставщиков для внешнего управления ключами на уровне предприятия;
- функции симметричного и асимметричного шифрования;
- использование представлений каталога для доступа к метаданным безопасности.

Перец - это значение, которое добавляется к хешированному паролю, который был посолен. Перец, как и соль – это набор символов, которые добавляются к паролю перед хешированием. В отличие от соли, перец существует в единственном виде и хранится в секрете, отдельно от солей и хешей. Он добавляет еще один слой безопасности, но если злоумышленник сможет получить к нему доступ, то ценность перца стремится к нулю, позволяя атакующему приступить к вычислению новых словарей и таблиц.

Шифрование на уровне приложений.

При шифровании на уровне приложений процесс шифрования данных завершается приложением, которое использовалось для создания или изменения данных, которые должны быть зашифрованы.

Преимущества. 1. шифрование на уровне приложений может упростить процесс шифрования, используемый компанией. 2. касается общей темы воровства. Учитывая, что данные шифруются перед записью на сервер, хакеру потребуется доступ к содержимому базы данных, а также к приложениям, которые использовались для шифрования и дешифрования содержимого базы данных, чтобы расшифровать конфиденциальные данные.

Недостатки. 1) используемые приложения, необходимо будет модифицировать для шифрования самих данных. Это может отнять значительное количество времени и других ресурсов; 2) шифрование на уровне приложения может ограничивать производительность базы данных; 3) сложность управления ключами возрастает, поскольку несколько разных приложений должны иметь полномочия и доступ для шифрования данных и записи их в базу данных.

Риски шифрования базы данных

Шифрование в базах данных MS SQL Server

Рассмотрим:

• модель шифрования SQL Server, включая иерархию шифрования SQL Server и концепции серверных сертификатов и ключей шифрования баз данных;

• расширяемое управление ключами, позволяющее использовать аппаратные модули защиты от независимых поставщиков для внешнего управления ключами на уровне предприятия;

• прозрачное шифрование данных, позволяющее незаметно шифровать всю базу данных сразу, не затрагивая интерфейсных приложений и приложений среднего слоя;

• использование представлений каталога для доступа к метаданным безопасности.

• функции симметричного и асимметричного шифрования;

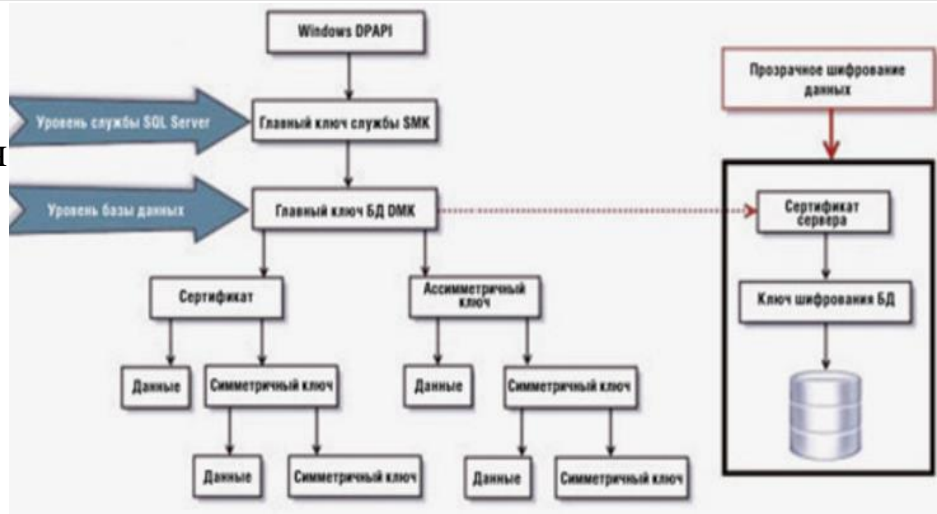
Мастер-ключ службы (SMK) – это ключ верхнего уровня, прародитель всех ключей SQL Server. На каждом экземпляре SQL Server определѐн только один SMK. Ключ SMK защищѐн интерфейсом Windows Data Protection API (DPAPI) и используется для защиты ключей следующего уровня - мастер-ключей базы данных (DMK). SMK автоматически создается SQL Server при первой необходимости.

Мастер-ключи баз данных (DMK) используются для шифрования симметричных ключей, асимметричных ключей и сертификатов. Каждая база данных имеет один DMK, определенный для нее.

Таблица Уровни шифрования ключей SQL Server и ANSI X9.17

Уровень SQL Server	Уровень ANSI X9.17	Описание
SMK	Главный ключ	SMK — ключ верхнего уровня, используемый для шифрования DMK. SMK шифруется с применением Windows DPAPI
DMK	Ключ шифрования ключей	DMK — симметричный ключ, используемый для шифрования симметричного ключа, асимметричного ключа и сертификата. Для каждой базы данных может быть определен только один DMK
Симметричные ключи, асимметричные ключи и сертификаты	Ключ данных	Симметричные ключи, асимметричные ключи и сертификаты используются для шифрования данных

Ключ шифрования - это специальный симметричный ключ, используемый для шифрования всей базы данных сразу.



Мастер-ключ службы

SQL Server 2008 включает следующие операторы T-SQL для изменения, резервирования и уничтожения SMK.

Alter – позволяет изменять или регенерировать SMK. Этот оператор может быть использован для изменения SMK и автоматической дешифрации и перешифровки всей иерархии ключей шифрования.

Backup – выполняет резервное копирование вашего SMK в файл. SMK шифруется перед резервным копированием и хранится в зашифрованном формате. Вы должны указать пароль для шифрования резервной копии SMK.

Restore – восстанавливает SMK из файла. Подобно оператора Alter этот оператор регенерирует всю иерархию ключей шифрования.

Мастер-ключи базы данных

Иерархия ключей шифрования SQL Server предусматривает по одному DMK для каждой базы данных. DMK непосредственно шифрует симметричные ключи и сертификаты, которые могут быть использованы для шифрования симметричных ключей. Симметричные ключи, в свою очередь, используются для шифрования других симметричных ключей и данных.

В отличии от SMK, который генерируется автоматически, как только в нем возникает потребность, DMK должен быть создан явно оператором Create. Как видите существует значительно много операторов работы с этим сертификатом.

Create – создает DMK.

Alter – позволяет регенерировать DMK или изменить способ защиты DMK. При этом все ключи, которые он защищает будут дешифрованы и перешифрованы заново.

Drop – удаляет DMK. Если какие-либо ключи уже зашифрованы этим DMK, то операция не выполнится.

Backup – резервирует DMK в файл.

Restore – восстанавливает DMK из файла.

Open – открывает DMK, чтобы его можно было использовать для шифровки и дешифрации.

Close – закрывает DMK, который был явно открыт посредством Open Master Key после завершения шифрования и дешифрации.

Асимметричные ключи

Система шифрования в SQL Server предоставляет поддержку асимметричных ключей, которые в действительности состоят из пары ключей шифрования: открытого и закрытого (секретного). Секретный ключ может иметь длину 512, 1024 или 2048 бит. Нужно учесть, что есть ограничение размера данных, с которыми может справиться асимметричное шифрование. В частности, алгоритм RSA_1024 может зашифровать значение varchar максимальной длины в 117 символов или значение nvarchar максимальной длины 58 символов.

SQL Server предоставляет следующие операторы для управления асимметричными ключами.

Create – позволяет сгенерировать новую пару асимметричных ключей (открытый/закрытый), импортировать такую пару из файла либо импортировать открытый ключ из сборки .NET. Этот оператор требует наличия привилегии CREATE ASYMMETRIC KEY в базе данных.

Alter – позволяет модифицировать свойства существующего асимметричного ключа.

Drop – удаляет асимметричный ключ из базы.

Операторы *Alter* и *Drop* требуют привилегии *CONTROL* для асимметричного ключа.

Кроме того есть две встроенные функции T-SQL для шифрования и дешифрации данных с помощью асимметричного ключа.

EncryptByAsymKey – эта функция требует применения номера ID пары асимметричного ключа, полученного от функции AsymKey_ID.

DecryptByAsymKey – дешифрует данные, которые были ранее зашифрованы посредством *EncryptByAsymKey*.

Сертификаты

Сертификат – это пара ключей открытый/секретный, которая содержит дополнительные данные, описывающие сертификат. Дополнительные данные включают дату начала, дату истечения срока действия и субъект сертификата. В отличии от асимметричных ключей SQL Server, сертификаты могут резервироваться в файлы и восстанавливаться из них. Сертификаты подписываются издателем сертификатов, которым часто бывает независимая компания, хотя SQL Server также может генерировать и автоподписанные сертификаты. SQL Server поддерживает сертификаты, которые следуют требованиям стандарта International Telecommunication Union Telecommunication Standartization Sector (ITU-T) X.509.

Microsoft рекомендует, чтобы сертификаты, как и асиметричные ключи, применялись для шифрования симметричных ключей, а симметричные ключи - для шифрования данных.

Симметричные ключи

Симметричные ключи находятся внизу иерархии ключей шифрования SQL Server. Симметричный ключ используется для шифрования других симметричных ключей или данных. Поскольку шифрование симметричным ключом осуществляется намного быстрее, чем асимметричным, и не страдает от ограничения длины, как в реализации асимметричных ключей SQL Server, Microsoft рекомендует шифровать данные исключительно симметричными ключами.

В то время как асимметричное шифрование требует двух ключей (пара открытый/секретный), для симметричного шифрования необходим единственный ключ, который используется для шифрования, и для расшифровки данных. Симметричное шифрование выполняется в соответствии с блочным алгоритмом, который шифрует данные блоками постоянной длины, и потоковым алгоритмом, шифрующим данные в непрерывном потоке. Алгоритмы блочного шифрования имеют набор длин ключей и размеров блоков, показанные на слайде.

В SQL Server доступны следующие операторы для управления симметричными ключами:

CREATE – создает симметричный ключ для использования в шифровании. Симметричные ключи могут быть зашифрованы сертификатами, асимметричными ключами, паролями или даже другими симметричными ключами.

ALTER – позволяет изменять метод защиты симметричных ключей.

DROP – удаляет симметричный ключ из базы данных. Симметричные ключи не могут быть уничтожены, пока они открыты.

OPEN – открывает и дешифрует симметричный ключ для использования

CLOSE – закрывает ранее открытый симметричный ключ.

CLOSE ALL – закрывает все в данные момент открытые симметричные ключи в текущем сеансе.

Можно создавать временные симметричные ключи, снабжая имя ключа префиксом - знаком # в операторе CREATE SYMMETRIC KEY.

Чтобы шифровать и дешифровать данные, применяют функции *EncryptByKey* и *DecryptByKey*.