САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Безопасность систем баз данных

Лекция 11
Шифрование

Агафонов Антон Александрович
к.т.н., доцент кафедры ГИиИБ

Самара

План лекции

- Шифрование
  - Шифрование базы данных (at rest)
  - Шифрование подвижных данных (in transit)

- Шифрование в СУБД
  - MySQL
  - PostgreSQL
  - MongoDB

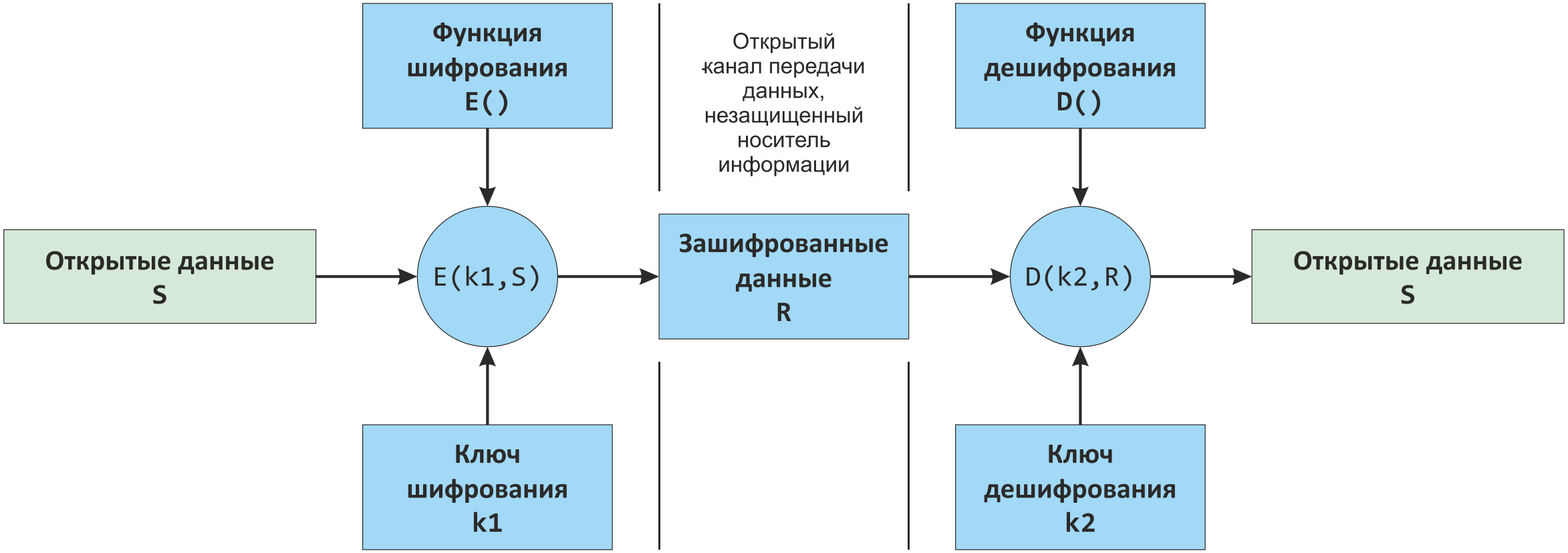
Шифрование

Шифрование – это процесс преобразования открытых данных с использованием специального алгоритма, после чего эти данные не могут быть восстановлены к исходному виду без ключа дешифрования.

Шифрование базы данных (неподвижных данных, at rest) — использование технологии шифрования для преобразования информации, хранящейся в базе данных, в шифротекст, что делает её прочтение невозможным для лиц, не обладающих ключами шифрования.

Шифрование «подвижных» данных (in transit)— шифрование данные при их передаче по сети между БД и любыми клиентами и приложениями.

Процесс шифрования



Виды шифрования БД

Виды шифрования БД:

- шифрование на уровне хранилища;
- шифрование на уровне базы данных;
- шифрование на уровне приложения.

Шифрование на уровне хранилища

Такой вид шифрования баз данных также называют **«прозрачным»** (Transparent Database Encryption, TDE).

Прозрачное шифрование данных используется для шифрования всей базы данных, что, следовательно, подразумевает шифрование «неактивных» данных. Под «неактивными» данными понимаются данные, которые в настоящее время не редактируются или не перемещаются по сети. Данные шифруются перед записью на диск и дешифруются во время чтения в память.

- ▲ Шифрование и дешифрование выполняются «прозрачно» для приложений, то есть не требуется их модификация. Также при использовании такого вида шифрования кроме непосредственно файлов базы данных также шифруются её резервные копии.
- ▼ Данный вид шифрования не обеспечивает сохранность информации при передаче по каналам связи или во время использования. Соответственно требуется дополнительные меры для шифрования данных на транспортном уровне.

К этому же виду шифрования также можно отнести шифрование на уровне файловой системы, например, с использованием системы шифрования данных EFS (Encrypting File System), реализованной в ОС Windows.

Шифрование на уровне базы данных

Шифрование на уровне базы данных используется для шифрования значений отдельных столбцов (Column-Level Encryption).

Данные шифруются и дешифруются непосредственно в момент их использования. Т.е. в базу данных помещаются уже зашифрованные значения, сама же база данных записывается на диск без дальнейшего шифрования.

- ▲ Возможность шифрования отдельных столбцов, что повышает гибкость шифрования в целом.
- ▲ Для каждого зашифрованного столбца можно использовать уникальный ключ шифрования, что повышает надежность защиты конфиденциальных данных.
- ▲ Данные не расшифровываются, пока не придет время их использовать, то есть данные, загруженные из хранилища в память, зашифрованы.
- ▼ Необходимость изменения схемы данных, так как все зашифрованные данные хранятся в двоичном виде.
- ▼ Снижается общая производительность СУБД. Во-первых, из-за дополнительной обработки при шифровании и расшифровке данных. Во-вторых, бо́льшего времени требует просмотр таблицы, поскольку индексы по зашифрованным столбцам не могут быть использованы.

Шифрование на уровне приложения

В шифровании на уровне приложений процесс шифрования осуществляется приложением, которое создаёт или изменяет данные, то есть шифрование данных происходит перед их записью в базу. Такой подход является весьма гибким, так как позволяет настроить процесс шифрования для каждого пользователя на основе информации о его правах в приложении.

- ▲ Нет необходимости использовать дополнительное решение для защиты данных при передаче по каналам связи, так как они отправляются уже зашифрованными.
- ▲ Кража конфиденциальной информации становится сложнее, так как злоумышленник должен иметь доступ не только к данным, но и к приложению, чтобы расшифровать данные.
- ▼ Для реализации шифрования на уровне приложений необходимо внесение изменений как в приложение, так и в базу данных.
- ▼ Могут возникнуть проблемы с производительностью БД, у которой пропадает возможность индексирования и поиска по зашифрованным данным.
- ▼ Сложность управления ключами в системе с таким шифрованием, т.к. несколько приложений могут использовать БД, и, следовательно, ключи хранятся во многих различных местах

Риски шифрования данных

- **Ненадлежащее управление ключами**. Если управление ключами шифрования не происходит в «изолированной системе», возникает риск того, что неблагонадежные сотрудники могут иметь возможность расшифровать конфиденциальные данные с помощью ключей, к которым они получат доступ.
- **Утрата ключа**. Утеря ключ порождает риск потери данных, так как расшифровка без ключей практически невозможна.

Шифрование подвижных данных

В процессе передачи данных между клиентом и сервером существует возможность перехвата нешифрованных данных злоумышленником. Такая уязвимость потенциально может эксплуатироваться следующими типами атак:

- **Подслушивание (Eavesdropping).** Атака на сетевом уровне, которая сфокусирована на перехвате пакетов, передаваемых по сети, и чтении их содержимого в поисках интересующей информации.
- **Посредник (Man-in-the-Middle).** Атака на сетевом уровне, в ходе которой злоумышленник встраивается в канал связи между клиентом и сервером и тайно ретранслирует трафик имея возможность анализировать и изменять содержимое пакетов.

Использование протоколов шифрования «подвижных» данных:

- посредством установления шифрованного соединения между клиентом и сервером, при этом СУБД и клиент должны поддерживать шифрованные соединения;
- с помощью организации защищенного туннеля между клиентом и сервером, такой подход применим для любых СУБД, т.к. реализуется сторонними программными средствами.

Использование шифрованного соединения

Суть использования шифрованного соединения заключается в шифровании и дешифровании сетевого траффика на концах устанавливаемого соединения, соответственно во время передачи по каналу связи данные находятся в зашифрованном виде.

SSL (Secure Sockets Layer) — криптографический протокол для безопасной связи. С версии 3.0 SSL заменили на TLS (Transport Layer Security). Хотя формально это два различных протокола, отличающиеся в некоторых деталях реализации, ввиду того что TLS является развитием SSL, в настоящее время под SSL чаще всего подразумевают TLS. Цель протокола — обеспечить защищенную передачу данных.

При этом для аутентификации используются асимметричные алгоритмы шифрования (пара открытый — закрытый ключ), а для сохранения конфиденциальности, с целью повышения скорости обработки данных — симметричные (секретный ключ).

Использование шифрованного соединения

Процесс установления соединения между клиентом и сервером (рукопожатие - handshake):

1) При запуске клиентского приложения оно запрашивает информацию о сертификате у сервера, который в свою очередь высылает копию SSL-сертификата с открытым ключом.
2) Клиент проверяет подлинность сертификата, дату действия сертификата и в некоторых случаях наличие корневого сертификата, выданного авторизованным центром сертификации.
3) Если подлинность сертификата подтверждена, то клиент генерирует предварительный секрет (Pre-master Secret) сессии на основе открытого ключа, полученного от сервера, используя максимально высокий уровень шифрования, который поддерживают обе стороны.
4) Сервер расшифровывает предварительный секрет с помощью своего закрытого ключа и использует его для вычисления общего секрета (Master Secret), используя определенный вид шифрования (два самых распространённых из них — RSA и Диффи-Хеллман).

Теперь обе стороны используют симметричный ключ, действительный только в рамках данной сессии (сессионный ключ) для шифрования пакетов. После завершения сессии ключ уничтожается, а при следующем установлении соединения описанный процесс запускается сначала.

Использование защищённых туннелей

Такой подход позволяет обеспечить должный уровень защиты сетевого трафика для клиентов, не поддерживающих SSL.

Суть этого подхода заключается в организации защищенного туннеля между определенными портами клиента и сервера. Со стороны клиента это может быть произвольный свободный порт (например, 63333), со стороны сервера, порт, который слушает сервер БД (гипотетически это также любой свободный порт, но для различных СУБД есть значения по умолчанию: 3306 для MySQL, 5432 для PostgreSQL).

Для организации тунеллирования могут использоваться различные службы, например, ssh, stunnel, ipseq и т.д. Различаются эти протоколы как-правило тем, на каком уровне TCP/IP они работают и, соответственно, какие возможности предоставляют.

Для клиента подключение к удаленному серверу БД будет выглядеть как локальное подключение к серверу БД через порт 63333. Хотя в таком варианте не производится никакого шифрования передаваемых данных, но т.к. открытые данные передаются по защищенному туннелю, обеспечивается защита трафика. Незащищенными участками маршрута трафика будут участки от сервера БД до сервера службы тунеллирования и от клиента службы тунеллирования до клиента БД, но с этим не связано никаких дополнительных рисков, так как эти службы работают на одном компьютере.

MySQL. Шифрование на уровне данных

В MySQL поддерживается шифрование отдельных полей данных на уровне сервера. Для шифрования / расшифровывания данных предназначен ряд функций, в т.ч.:

Функции шифрования:

- `AES_ENCRYPT()` — использует алгоритм AES;
- `ASYMMETRIC_ENCRYPT()` — использует асимметричный ключ для шифрования данных.

Функции дешифрования :

- `AES_DECRYPT()` — использует алгоритм AES;
- `ASYMMETRIC_DECRYPT()` — использует асимметричный ключ для расшифровки данных.

Сервисные функции:

- `CREATE_ASYMMETRIC_PRIV_KEY()` — генерация закрытого ключа;
- `CREATE_ASYMMETRIC_PUB_KEY()` — генерация открытого ключа;
- `MD5()` — генерация MD-5 хэша;
- `SHA1(), SHA2()` — генерация SHA хэшей.

MySQL. Шифрование на уровне полей данных

```
CREATE TABLE users (
    username VARCHAR(50) NOT NULL,
    password BLOB NOT NULL
);

INSERT INTO users (username, password)
VALUES
('admin', aes_encrypt('33NC!DEgtFc8g3!2', 'Strong password')),
('joe', aes_encrypt('123', 'Strong password')),
('jane', aes_encrypt('qwerty', 'Strong password')),
('sam', 'samsam');
```

Просмотр табличных данных

```
SELECT username, CAST(password AS CHAR), HEX(password)
FROM users;
```

	username	CAST(password AS CHAR)	HEX(password)
	admin	NULL	F9F1976A13B0178955…
	joe	NULL	892682D81C2CA09DA…
	jane	NULL	618CB308A5A591BA4…
	sam	samsam	73616D73616D

Просмотр дешифрованных данных

```
SELECT username,
       CAST(aes_decrypt(password, 'Strong password') AS CHAR) AS text_password
FROM users
WHERE username = 'jane';
```

	username	text_password
	jane	qwerty

Просмотр дешифрованных данных (неправильный ключ)

```
SELECT username,
       CAST(aes_decrypt(password, 'Weak password') AS CHAR) AS text_password
FROM users
WHERE username = 'jane';
```

	username	text_password
	jane	NULL

MySQL. Прозрачное шифрование данных

В MySQL поддерживается возможность прозрачного шифрования всей базы данных. Реализовано AES-256 шифрование для табличных пространств InnoDB.

Используется двухуровневая схема ключей шифрования, состоящая из главного ключа (мастер-ключа) шифрования и ключей табличного пространства.

- Когда табличное пространство зашифровано, ключ табличного пространства шифруется мастер-ключом и хранится в заголовке табличного пространства.
- Когда необходимо получить доступ к зашифрованным данным табличного пространства, InnoDB использует мастер-ключ шифрования для расшифровки ключа табличного пространства.
- Ключ табличного пространства не меняется, а главный ключ шифрования может быть изменен при необходимости (например, при подозрении, что мастер-ключ скомпрометирован) – это называют «ротацией мастер-ключа».

Ротация мастер-ключа выполняется как атомарная операция на уровне экземпляра сервера.

Каждый раз, когда мастер-ключ меняется, все ключи табличных пространств в экземпляре MySQL заново зашифровываются и сохраняются обратно в заголовки соответствующих табличных пространств.

Шифрование с помощью нового мастер-ключа должно быть успешно завершено для всех ключей табличных пространств, если этот процесс прерывается из-за сбоя сервера, InnoDB откатывает операцию при перезагрузке сервера.

При ротации мастер-ключа изменяется только главный ключ шифрования и заново зашифровываются ключи табличных пространств. Данные, хранимые в соответствующих табличных пространствах, не расшифровываются и повторно не зашифровываются, т.к. непосредственно ключ табличного пространства не меняется.

Защита соединений в MySQL

По умолчанию MySQL сервер всегда включает поддержку SSL-подключений, но при этом не требуется обязательного SSL-подключения для клиентов. Клиенты могут выбирать соединение с SSL или без него, при подключении это управляется параметром `ssl-mode`.

Для указания серверу, что все подключения должны выполняться с использованием SSL, используется системная переменная `require_secure_transport`, которая по умолчанию имеет значение `OFF`.

Для генерации ключей и сертификатов используется утилита `mysql_ssl_rsa_setup`, которая поставляется вместе с MySQL. Сгенерированные сертификаты и ключи сохраняются в виде PEM-файлов в директории данных MySQL . При каждом старте сервер анализирует наличие ключей в папке данных и, если они найдены, то автоматически включает поддержку SSL-подключений.

PostgreSQL. Шифрование на уровне полей данных

Модуль pgcrypto предоставляет криптографические функции для PostgreSQL.

Данный модуль поддерживает как функции шифрования на основе PGP, так и низкоуровневые функции шифрования.

PGP (Pretty Good Privacy) — библиотека функций, позволяющая выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, представленной в электронном виде, в том числе прозрачное шифрование данных на запоминающих устройствах, например, на жёстком диске.

Шифрование PGP осуществляется последовательно хешированием, сжатием данных, шифрованием с симметричным ключом, и, наконец, шифрованием с открытым ключом, причём каждый этап может осуществляться одним из нескольких поддерживаемых алгоритмов.

Симметричное шифрование производится с использованием одного из симметричных алгоритмов на сеансовом ключе. Сеансовый ключ генерируется с использованием криптостойкого генератора псевдослучайных чисел. Сеансовый ключ зашифровывается открытым ключом получателя с использованием алгоритмов RSA или Elgamal.

Функции шифрования:

- `pgp_sym_encrypt` — шифрует данные симметричным ключом PGP;
- `pgp_pub_encrypt` — зашифровывает данные открытым ключом PGP.

Функции дешифрования :

- `pgp_sym_decrypt` — расшифровывает сообщение, зашифрованное симметричным ключом PGP;
- `pgp_pub_decrypt` — расшифровывает сообщение, зашифрованное открытым ключом. Принимает в качестве параметра закрытый ключ, соответствующий открытому ключу, применяющемуся при шифровании.

Сервисные функции:

- `digest` — вычисляет двоичный хеш данных используя заданный алгоритм (`md5`, `sha1`, `sha224`, `sha256`, `sha384` и `sha512`);
- `crypt` — выполняет хеширование паролей используя заданный алгоритм (`bf`, `md5`, `xdes`, `des`);
- `gen_salt` — вычисляет значение соли для функции `crypt`.

PostgreSQL. Шифрование на уровне полей данных

```
CREATE TABLE users_data (
    username VARCHAR(50) PRIMARY KEY,
    data bytea
);

INSERT INTO users_data (username, data)
VALUES
('admin', pgp_sym_encrypt('admin info', 'strong key')),
('joe', pgp_sym_encrypt('joe info', 'strong key')),
('jane', pgp_sym_encrypt('jane info', 'strong key')),
('sam', pgp_sym_encrypt('sam info', 'strong key'));
```

Просмотр табличных данных

```
SELECT username, encode(data, 'hex')
FROM users_data;
```

	username	encode
	admin	c30d040703029245b44c…
	joe	c30d040703020fb4100…
	jane	c30d0407030235590cfa…
	sam	c30d0407030266ff96eb…

Просмотр дешифрованных данных

```
SELECT username, pgp_sym_decrypt(data, 'strong key')
FROM users_data
WHERE username = 'jane';
```

	username	pgp_sym_decrypt
	jane	jane info

Просмотр дешифрованных данных (неправильный ключ)

```
SELECT username, pgp_sym_decrypt(data, 'wrong key')
FROM users_data
WHERE username = 'jane';
```

ERROR: ОШИБКА: Wrong key or corrupt data

PostgreSQL. Шифрование паролей

```
CREATE TABLE users (
    username VARCHAR(50) PRIMARY KEY,
    password TEXT
);

INSERT INTO users (username, password)
VALUES
('admin', crypt('33NC!DEgtFc8g3!2', gen_salt('md5'))),
('joe', crypt('qwerty', gen_salt('md5'))),
('jane', crypt('qwerty', gen_salt('md5'))),
('sam', crypt('samsam', gen_salt('md5')));
```


Просмотр табличных данных

```
SELECT username, password
FROM users;
```

	username	password
	admin	1oEbgx9qm$0eM8LWnlyN8ITQgBfRd2o0
	joe	1txB6MCiD$AVq8j491q1be8tBGCW17j.
	jane	1YxXGY0E3$hSLteQ7jFwbWFA0Xqbsnp/
	sam	1JuJT3BBM$UGDccGzqfvP68elJ68.AD1

Проверка пароля

```
SELECT username
FROM users
WHERE username = 'sam' AND password = crypt('samsam', password);
```

	username
	sam

Просмотр пароля (неправильный пароль)

```
SELECT username
FROM users
WHERE username = 'sam' AND password = crypt('qwerty', password);
```

	username

Защита соединений в PostgresSQL

В PostgresSQL встроена поддержка SSL для шифрования трафика между клиентом и сервером. Для использования этой возможности необходимо, чтобы и на сервере, и на клиенте была установлена криптографическая библиотека OpenSSL и включена поддержка SSL в конфигурационном файле postgresql.conf.

Запущенный сервер будет принимать как обычные, так и SSL-подключения по одному порту TCP. По умолчанию клиент выбирает режим подключения сам, но для подключений, у которых в конфигурационном файле pg_hba.conf указан тип соединения `hostssl`, всегда будут устанавливаться SSL-подключения.

Для генерации ключей и сертификатов используется библиотека OpenSSL.

MongoDB. Шифрование данных

Шифрование данных доступно в MongoDB Enterprise.

Процесс шифрования данных включает в себя следующие этапы:

- генерирование мастер-ключа;
- генерирование ключей для каждой базы данных;
- шифрование данных с помощью ключей базы данных;
- шифрование ключей базы данных с помощью мастер-ключа.

При использовании шифрования все файлы данных шифруются в файловой системе. Данные не шифруются только в памяти и во время передачи.

MongoDB. Шифрование данных

Ключи базы данных являются внутренними для сервера и записываются на диск только в зашифрованном формате. Мастер-ключ на диск не записывается.

Для управления мастер-ключом MongoDB поддерживает два варианта управления ключами:

- *Интеграция со сторонним устройством управления ключами через протокол совместного управлением ключами (Key Management Interoperability Protocol, KMIP).*
- Локальное управление ключами через файл ключа.

MongoDB. Шифрование подвижных данных

MongoDB поддерживает TLS/SSL (Transport Layer Security/Secure Sockets Layer) для шифрования всего сетевого трафика MongoDB. TLS/SSL гарантирует, что сетевой трафик MongoDB доступен для чтения только предполагаемому клиенту.

MongoDB использует библиотеки TLS/SSL операционных систем:

- Windows — Secure Channel (Schannel)
- Linux/BSD — OpenSSL
- macOS — Secure Transport

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
к.т.н., доцент кафедры ГИиИБ