

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
 «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

Лабораторная работа №7

по дисциплине «Основы построения защищённых баз данных»

Выполнил: обучающийся гр. ВКБ43

Ковалев Данил Петрович

Проверил:

Скляров Алексей Викторович

**Цель:** изучить технологии шифрования данных в СУБД, включая иерархию шифрования, использование симметричных и асимметричных ключей, а также прозрачное шифрование данных (TDE).

**Задание 1**. Включить шифрование соединений в ядре СУБД SQL Server и описать алгоритм.

Для включения шифрования соединений нужно создать сертификаты и выставить нужные параметры в SQL Server. Из документации Microsoft было подмечено, что для тестовых сред можно просто включить режим “forceencryption” в mssql.conf, после чего база данных сама сгенерирует сертификаты и подключит режим подключения – SSL. Конфигурация msssql.conf представлена на рисунке 1.

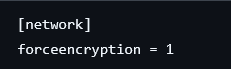


Рисунок 1 – конфигурация mssql.conf

Дальше нужно данный файл применить при запуске. В моем случае конфигурацией перед стартом занимается Docker compose. Код для создания mssql базы данных представлен на рисунке 2.



Рисунок 2 – код для создания контейнера с mssql

Проверим теперь, что шифрования соединений включено. Для этого была использована команда, которая представлена на рисунке 3.

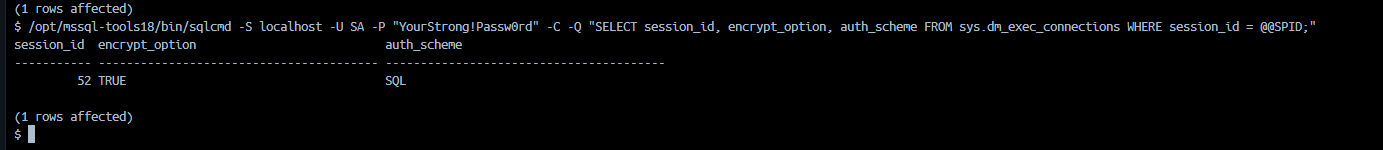


Рисунок 3 – успешность включения шифрования канала

**Задание 2**. Включить прозрачное шифрование данных и описать каждый шаг алгоритма.

В самом начале необходимо создать главный ключ для шифрования. В документации SQL Server сказано, что его можно создать подобным образом, как представлено ниже на рисунке 4. В зависимости от операционной системы присутствует различное поведение создание ключей, то есть где-то автоматически, а где-то вручную.

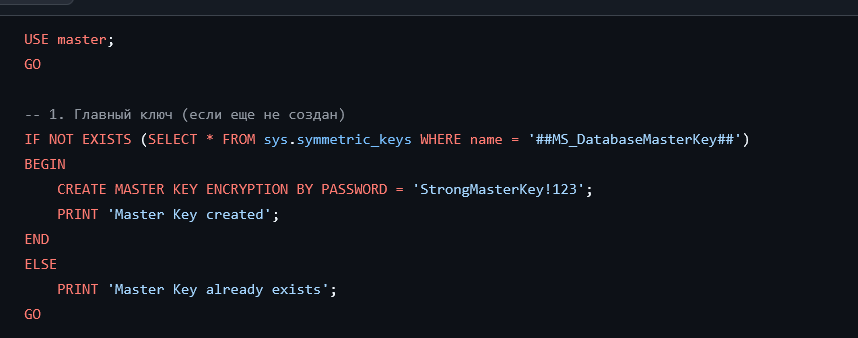


Рисунок 4 – создание главного ключа шифрования

Теперь создадим вручную сертификат. Был описан такой код, как представлено ниже на рисунке 5.

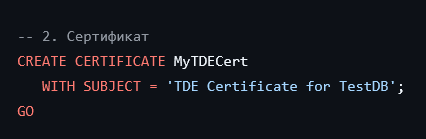


Рисунок 5 – создание сертификата

Теперь создадим тестовую базу данных, в случае которой будет работать наша лабораторная работа. Код представлен на рисунке 6.

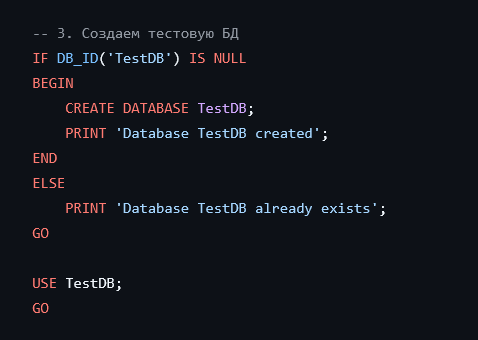


Рисунок 6 – создание базы данных для лабораторной работы

Создадим ключ шифования для базы данных и включим режим шифрования для базы данных, как представлено на рисунке 7.

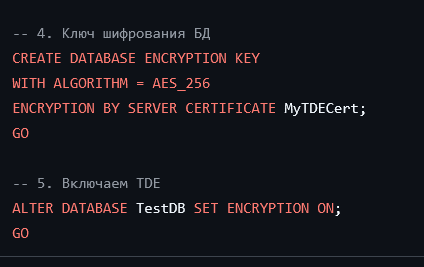


Рисунок 7 – создания ключа шифрования для базы данных и включение шифрования ядра

Проверим, что прозрачное шифрование – TDE – было удачно создано и примнено. Для этого используется команда, которая представлена на рисунке 8.

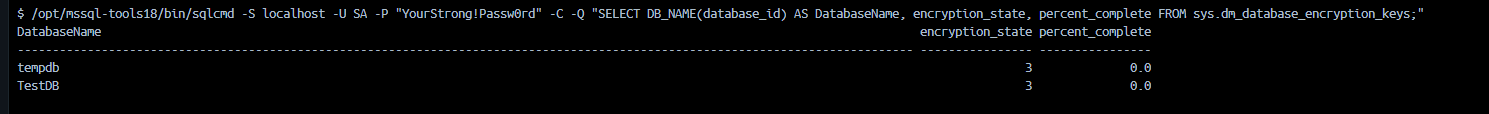


Рисунок 8 – успешность включения шифрования для базы данных

**Задание 3.** Создать таблицу в БД и реализовать процедуру (или процедуры) на языке T-SQL, которые предоставляют возможности внесения и получения данных, хранящихся в зашифрованном виде (SELECT вернет зашифрованные данные) с использованием EncryptByPassPhrase/DecryptByPassPhrase.

Для выполнения задания было написано несколько SQL скриптов, где есть описание таблиц и процедур. Для начала рассмотрим файл “2.sql”, в котором описано создание таблиц, хранящая данные, зашифрованные с помощью симметричного алгоритма. Были также разработаны процедуры для просмотра данных и вставки данных. Код можно рассмотреть более подробно на рисунке 9.

Говоря более подробно, процедура InsertEncryptedSecret нужна с той целью, чтобы вставлять данные и шифровать с помощью симметричного алгоритма. В нашем случае ключом является фраза – MySuperSecretPassPhrase!123. Процедура GetEncryptedSecrets нужна с той целью, чтобы просматривать данные в зашифрованном виде. Процедура DecryptSecret нужна для просмотра дешифрованных данных.

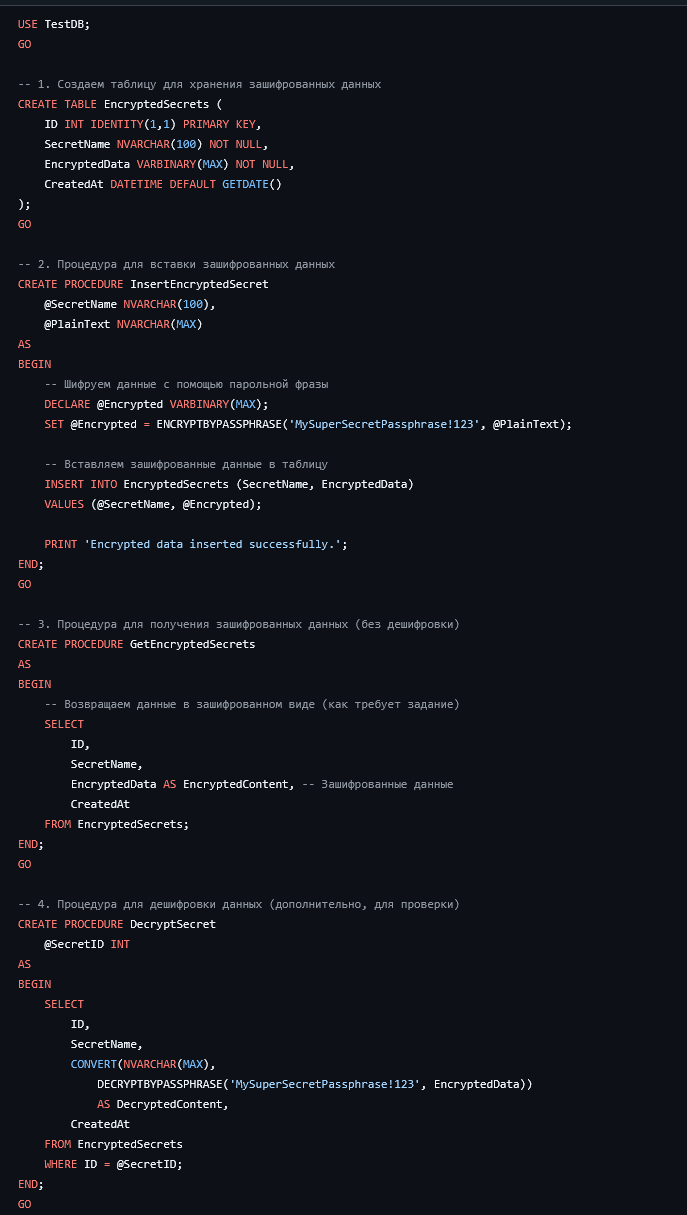


Рисунок 9 – код для создания таблиц и процедур

Попробуем теперь выполнить вставку данных в таблицу, используя процедуры, описанные до этого. Результат одного запроса видно на рисунке 10.

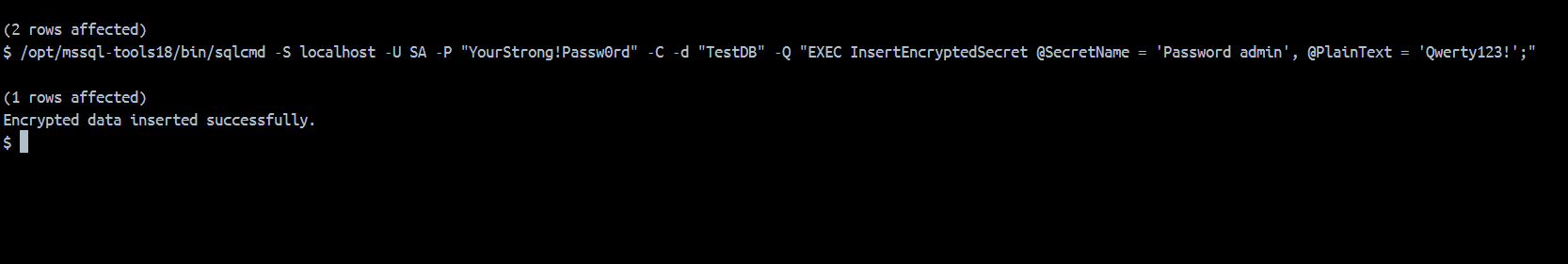


Рисунок 10 – вставка данных в зашифрованном виде

Просмотрим теперь данные в зашифрованном виде. Результат представлен на рисунке 11.

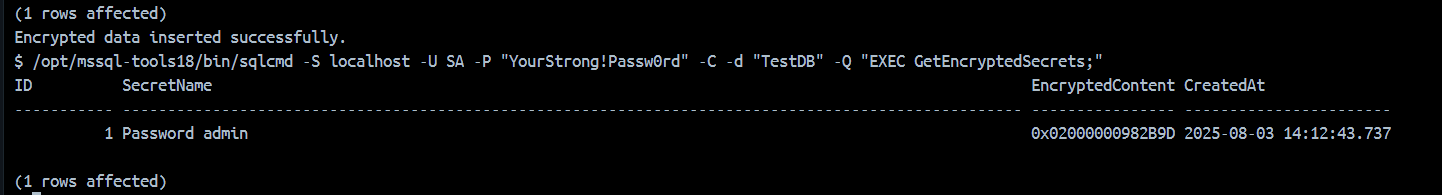


Рисунок 11 – вставка зашифрованных данных

Проверим теперь дешифровку, что она отлично работает. Результат представлен на рисунке 12.

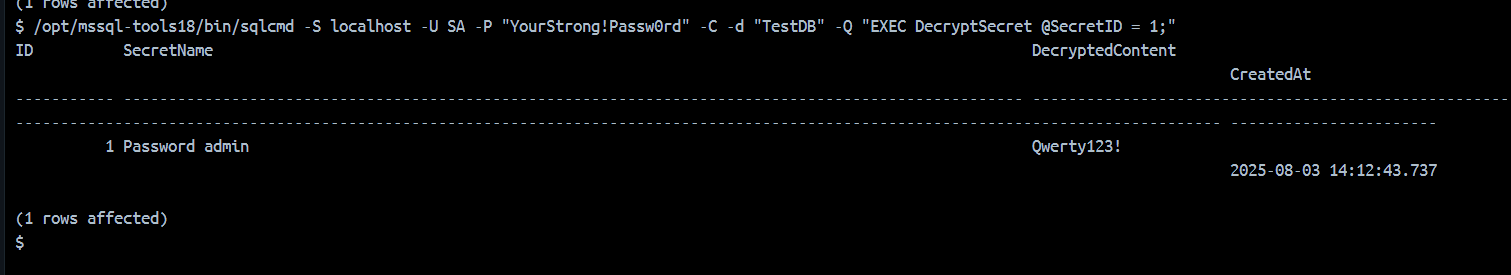


Рисунок 12 – результат удачной дешифрации

Попробуем теперь дешифровать, введя неправильный пароль. Результат представлен на рисунке 13. При неправильной дешифровке MS SQL возвращает просто NULL.

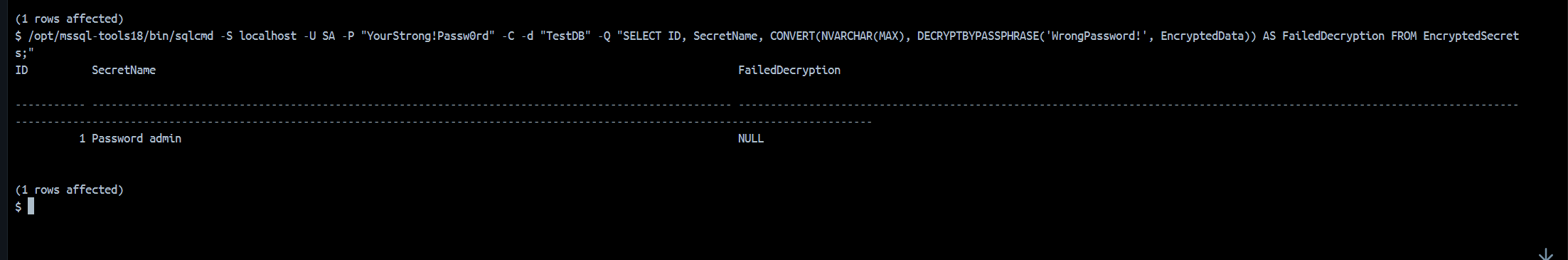


Рисунок 13 – провальная дешифрация с неправильным паролем

**Задание 4**. Продемонстрировать процесс шифрования данных с использованием симметричного ключа, зашифрованного с помощью ассиметричного ключа (написать скрипт):

Для выполнения задания был написан скрипт SQL, который создает симметричный и ассиметричные ключи для шифрования содержимого базы данных. Симметричный ключ у нас защищен ассиметричным ключом.

На рисунке 14 также описано создание базы данных. Также изображен пример заранее заготовленной вставки, которая прогоняется как миграция базы данных.

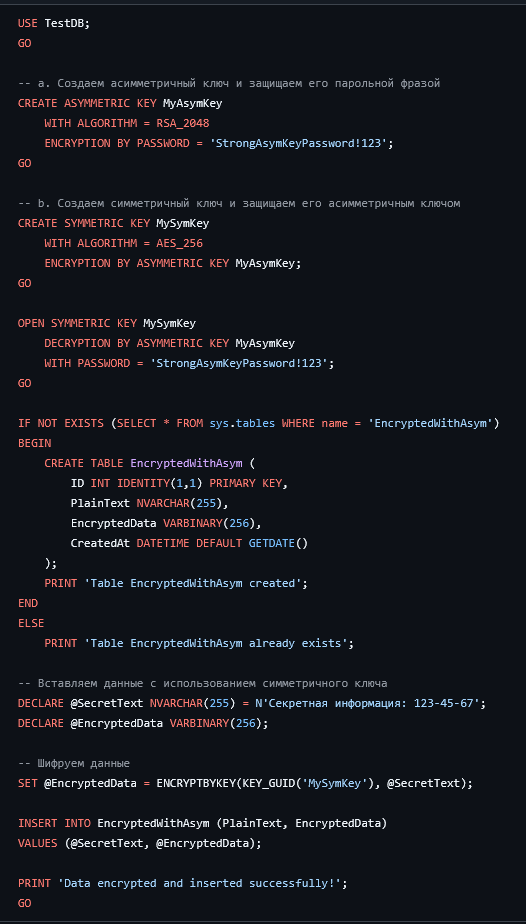


Рисунок 14 – создание таблицы для встваки зашифрованных данных

Попробуем теперь узнать какие у нас есть таблицы в базе данных, результат представлен на рисунке 15.

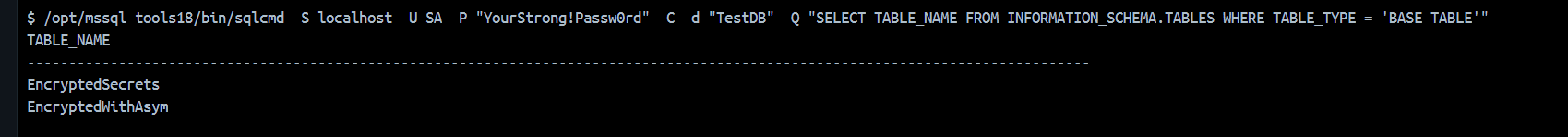


Рисунок 15 – таблицы базы данных, которые были созданы

Проверим, что ассиметричный ключ был удачно создан, результат представлен на рисунке 16.

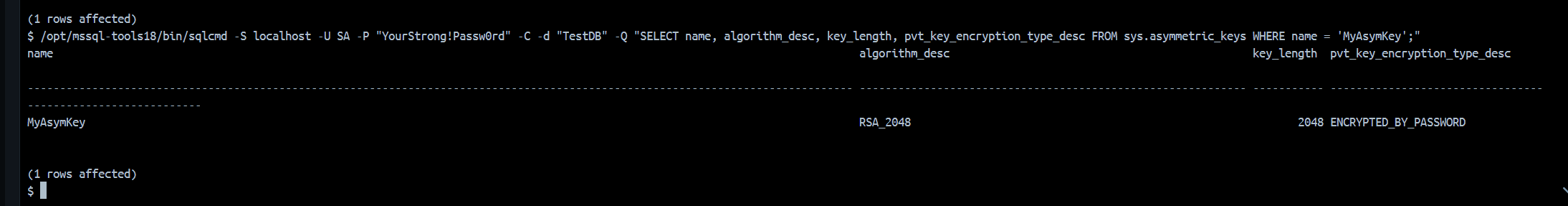


Рисунок 16 – проверка существования асимметричного ключа

Проверим теперь, что симметричный ключ удачно создан, результат представлен на рисунке 17.

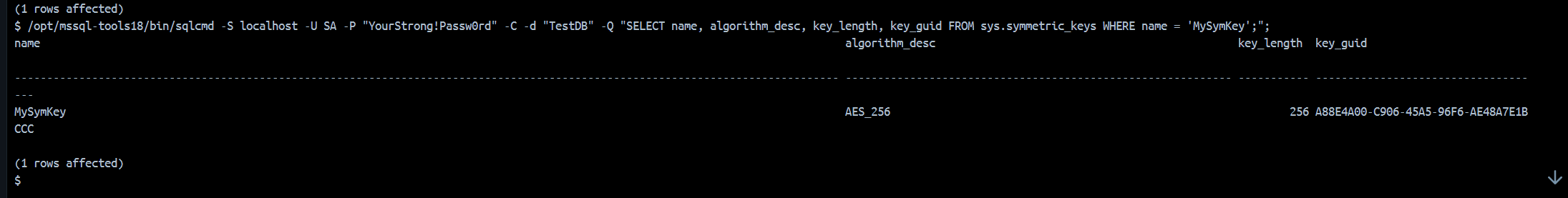


Рисунок 17 – проверка существования симметричного ключа

Так как каждый запрос у меня — это отдельная сессия, то мне придется теперь при каждом запросе открывать симметричный ключ для каких-нибудь изменений. Приложу пример запроса, который открывает симметричный ключ.

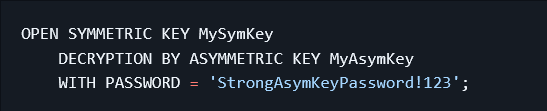


Рисунок 18 – открытие симметричного ключа

Проверим теперь, что ключ открыт и можно использовать его, для этого нужно выполнить запрос, который представлен на рисунке 19.

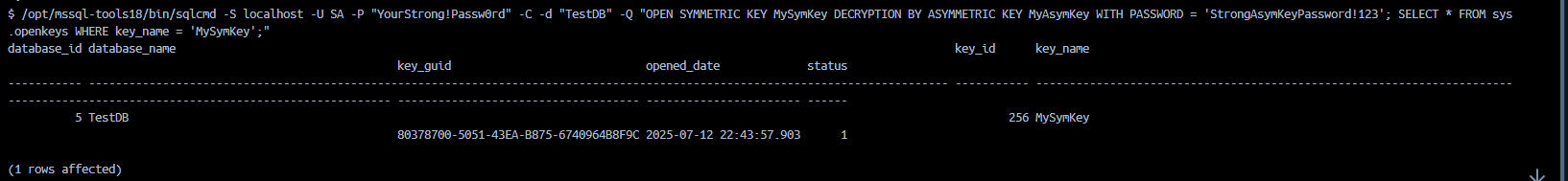


Рисунок 19 – проверка, что ключ открыт успешно

В базе данных TestDB уже были заготовлены данные на вставку через миграции, поэтому покажу результат как данные хранятся в зашифрованном виде. Результат представлен на рисунке 20.

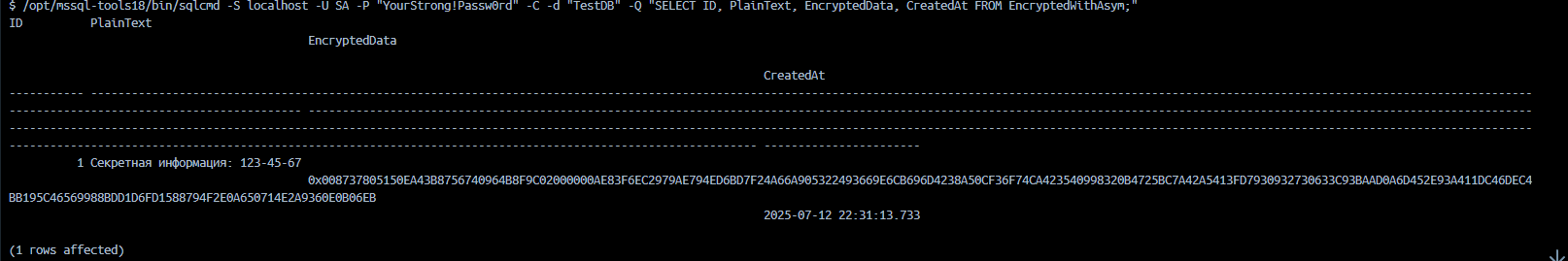


Рисунок 20 – данные, зашифрованные симметричным ключом, который подписан ассиметричным ключом

Теперь попробуем посмотреть дешифрованные данные, результат представлен на рисунке 21.

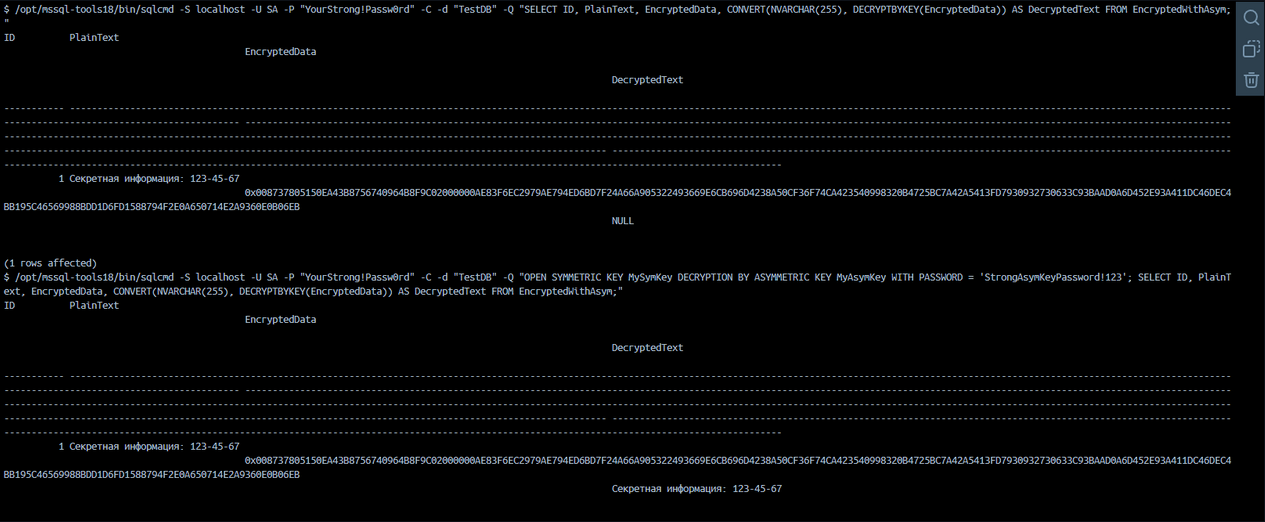


Рисунок 21 – дешифрованные данные, зашифрованные симметричным ключом, который подписан ассиметричным ключом