Secure Systems & Network Engineering

Advanced Security

Lab 6 SQL

Grebennikov Sergey

February 18, 2018

Contents

1 Standard Database Encryption

Installing

MS SQL Server 2017 Enterprise for students was chosen as SQL Server. In addition, it is required to install SQL Server Management Studio for convenient work with MS SQL Server (Figure 1).

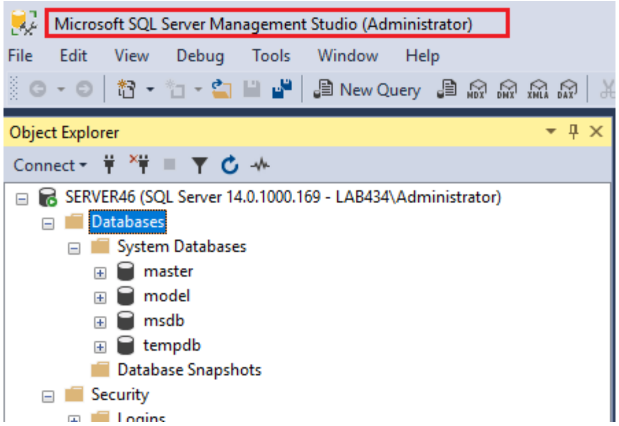
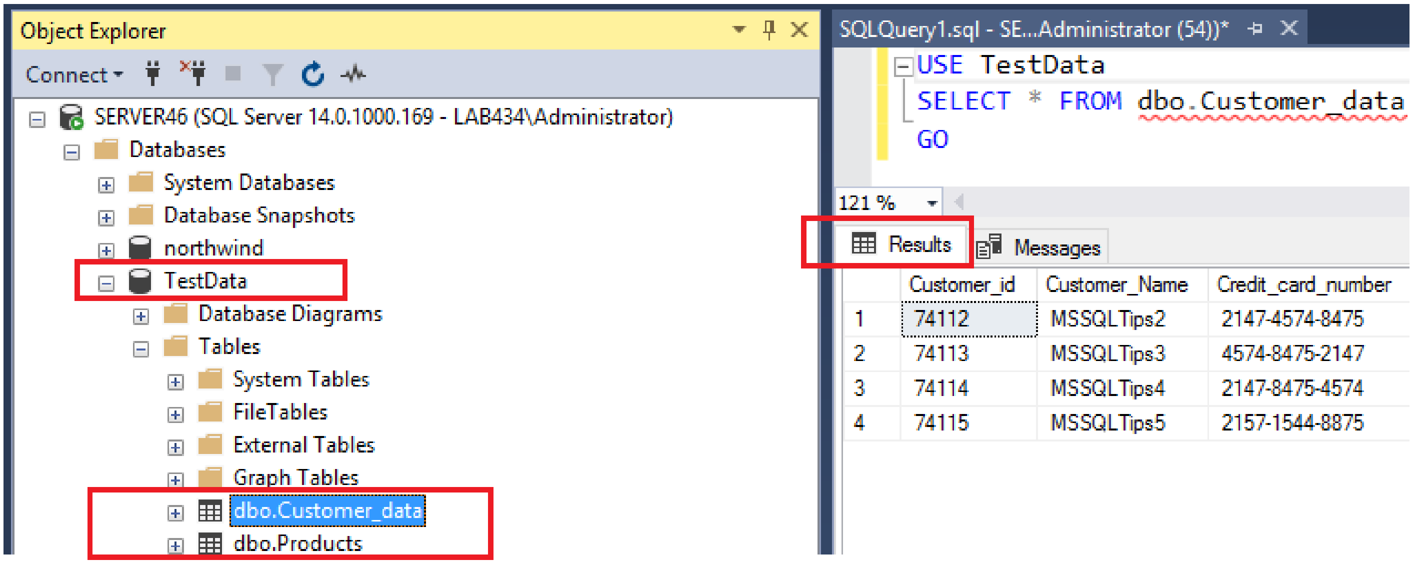

Microsoft SQL Server Management Studio (Administrator)
File Edit View Debug Tools Window Help
New Query
Object Explorer
Connect
SERVER46 (SQL Server 14.0.1000.169 - LAB434\Administrator)
Databases
System Databases
master
model
msdb
tempdb
Database Snapshots
Security
Logins

Figure 1: SQL Server Management Studio

Databases

Simple Database - TestData

To generate a simple database I created a new database and tables, and then inserted values into the tables (Figure 2).



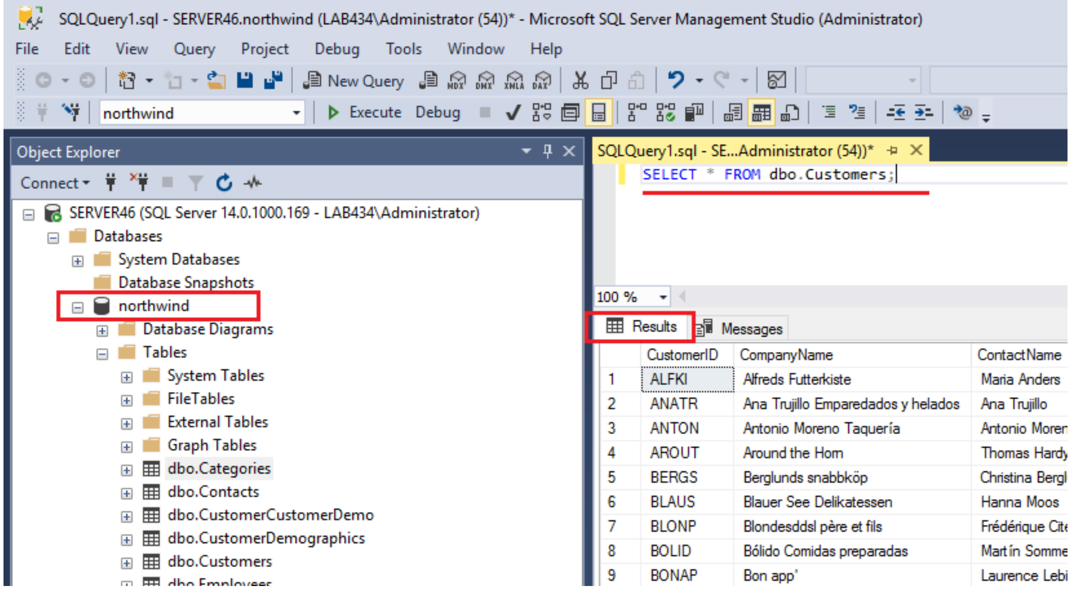
```
USE TestData
SELECT * FROM dbo.Customer_data
GO
```

	Customer_id	Customer_Name	Credit_card_number
1	74112	MSSQLTips2	2147-4574-8475
2	74113	MSSQLTips3	4574-8475-2147
3	74114	MSSQLTips4	2147-8475-4574
4	74115	MSSQLTips5	2157-1544-8875

Figure 2: TestData database

Large database - Northwind

To generate a large database I downloaded and installed the ”Northwind” database template (Figure 3).



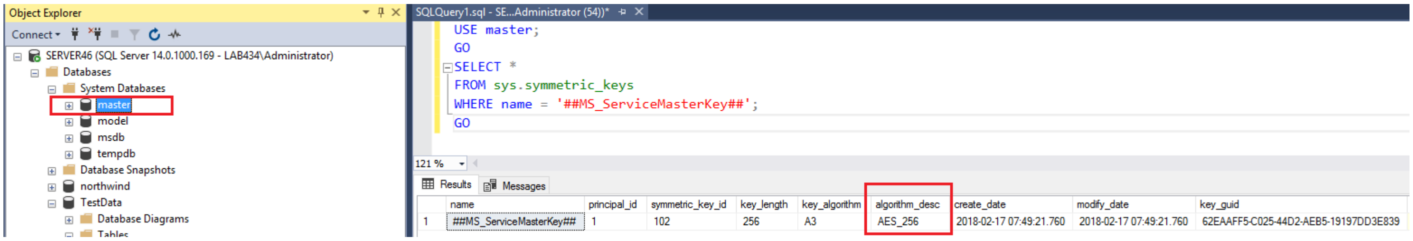
```
SELECT * FROM dbo.Customers;
```

	CustomerID	CompanyName	ContactName
1	ALFKI	Alfreds Futterkiste	Maria Anders
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo
3	ANTON	Antonio Moreno Taquería	Antonio Morer
4	AROUT	Around the Horn	Thomas Hardy
5	BERGS	Berglunds snabbköp	Christina Bergl
6	BLAUS	Blauer See Delikatessen	Hanna Moos
7	BLONP	Blondesddsl père et fils	Frédérique Cite
8	BOLID	Bólido Comidas preparadas	Martín Somme
9	BONAP	Bon app'	Laurence Lebi

Figure 3: Northwind database

Column level encryption

1. Check the existence of the SQL Server Service Master Key (Figure 4).



```
USE master;
GO
SELECT *
FROM sys.symmetric_keys
WHERE name = '##MS_ServiceMasterKey##';
GO
```

	name	principal_id	symmetric_key_id	key_length	key_algorithm	algorithm_desc	create_date	modify_date	key_guid
1	##MS_ServiceMasterKey##	1	102	256	A3	AES_256	2018-02-17 07:49:21.760	2018-02-17 07:49:21.760	62EAAFF5-C025-44D2-AEB5-19197DD3E839

Figure 4: SQL Server Service Master Key

2. Create the database Key

```
USE TestData;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Password123';
GO
```

3. Create a Self Signed SQL Server Certificate

```
CREATE CERTIFICATE Certificate1
WITH SUBJECT = 'Protect Data';
GO
```

4. SQL Server Symmetric Key

```
CREATE SYMMETRIC KEY SymmetricKey1
 WITH ALGORITHM = AES_128
 ENCRYPTION BY CERTIFICATE Certificate1;
GO
```

5. Schema changes. An Encrypted column can only be of datatype *varbinary* and since the column we want to encrypt is of datatype *varchar*, we have to create a new column and populate it with encrypted values.

```
ALTER TABLE Customer_data
ADD Credit_card_number_encrypt varbinary(MAX) NULL
GO
```

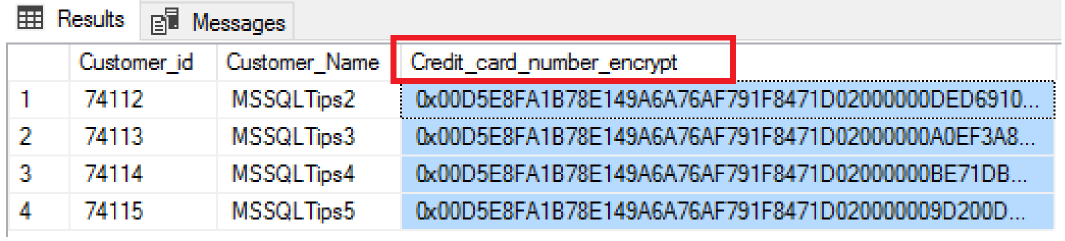
6. Encrypting the newly created column

```
-- Opens the symmetric key for use
OPEN SYMMETRIC KEY SymmetricKey1
DECRYPTION BY CERTIFICATE Certificate1;
GO
UPDATE Customer_data
SET Credit_card_number_encrypt = EncryptByKey
    (Key_GUID('SymmetricKey1'),Credit_card_number)
FROM dbo.Customer_data;
GO
-- Closes the symmetric key
CLOSE SYMMETRIC KEY SymmetricKey1;
GO
```

7. Remove a column

```
ALTER TABLE Customer_data
DROP COLUMN Credit_card_number;
GO
```

8. Result Figure 5



	Customer_id	Customer_Name	Credit_card_number_encrypt
1	74112	MSSQLTips2	0x00D5E8FA1B78E149A6A76AF791F8471D02000000DED6910...
2	74113	MSSQLTips3	0x00D5E8FA1B78E149A6A76AF791F8471D02000000A0EF3A8...
3	74114	MSSQLTips4	0x00D5E8FA1B78E149A6A76AF791F8471D02000000BE71DB...
4	74115	MSSQLTips5	0x00D5E8FA1B78E149A6A76AF791F8471D020000009D200D...

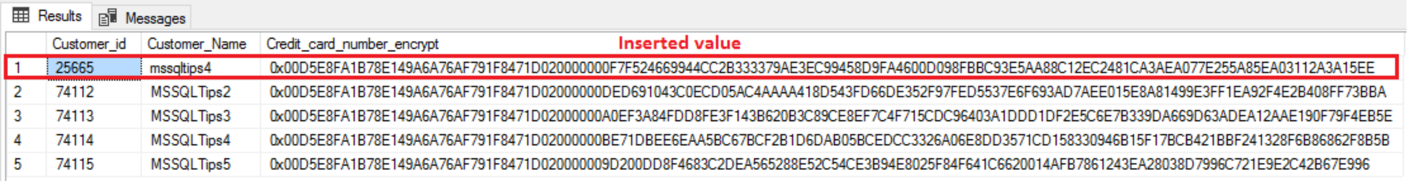
Figure 5: Example of the encrypted data

Insert encrypted values into the table

Script:

```
OPEN SYMMETRIC KEY SymmetricKey1
DECRYPTION BY CERTIFICATE Certificate1;
-- Performs the update of the record
INSERT INTO dbo.Customer_data (Customer_id, Customer_Name, Credit_card_number_encrypt)
VALUES (25665, 'mssqltips4', EncryptByKey( Key_GUID('SymmetricKey1'),
    CONVERT(varchar,'4545-58478-1245') ) );
GO
```

Result Figure 6



	Customer_id	Customer_Name	Credit_card_number_encrypt
1	25665	mssqltips4	0x00D5E8FA1B78E149A6A76AF791F8471D020000000F7F524669944CC2B333379AE3EC99458D9FA4600D098FBBC93E5AA88C12EC2481CA3AEA077E255A85EA03112A3A15EE
2	74112	MSSQLTips2	0x00D5E8FA1B78E149A6A76AF791F8471D02000000DED691043C0ECD05AC4AAAA418D543FD66DE352F97FED5537E6F693AD7AEE015E8A81499E3FF1EA92F4E2B408FF73BBA
3	74113	MSSQLTips3	0x00D5E8FA1B78E149A6A76AF791F8471D02000000A0EF3A84FDD8FE3F143B620B3C89CE8EF7C4F715CDC96403A1DDD1DF2E5C6E7B339DA669D63ADEA12AAE190F79F4EB5E
4	74114	MSSQLTips4	0x00D5E8FA1B78E149A6A76AF791F8471D02000000BE71DBEE6EAA5BC67BCF2B1D6DAB05BCEDCC3326A06E8DD3571CD158330946B15F17BCB421BBF241328F6B86862F8B5B
5	74115	MSSQLTips5	0x00D5E8FA1B78E149A6A76AF791F8471D020000009D200DD8F4683C2DEA565288E52C54CE3B94E8025F84F641C6620014AFB7861243EA28038D7996C721E9E2C42B67E996

Figure 6: Inserted value

Supported encryption algorithms

SQL Server allows administrators and developers to choose from among several algorithms, including DES, Triple DES, TRIPLE_DES_3KEY, RC2, RC4, 128-bit RC4, DESX, 128-bit AES, 192-bit AES, and 256-bit AES. But, beginning with SQL Server 2016, all algorithms other than AES_128, AES_192, and AES_256 are deprecated. To use older algorithms (not recommended) you must set the database to database compatibility level 120 or lower.

Performance comparison

Encryption	Time	Rows
Without encryption	1 min 30 sec	10 000
One column encryption	1 min 59 sec sec	10 000
Full table encryption	2 min 02 sec	10 000

Encryption did not affect the performance in my case.

Questions

a) What are the vulnerabilities (attack vectors) you could have with this approach? Can you get the plain text from encrypted data in a way?
Answer:
It is possible to implement the following attacks:

- *frequency analysis*: is a well-known attack that decrypts DTE-encrypted columns given an auxiliary dataset that is well-correlated with the plaintext column.

- *‘p-optimization*: is a new family of attacks that decrypts DTE-encrypted columns. The family is parameterized by the ‘p-norms and is based on combinatorial optimization techniques.
- *sorting attack* is an attack that decrypts OPE-encrypted columns. It is applicable to columns that are dense in the sense that every element of the message space appears in the encrypted column.
- *cumulative attack*: is a new attack we introduce that decrypts OPE-encrypted columns. This attack is applicable even to low-density columns and also makes use of combinatorial optimization techniques.

I can get the plain text from encrypted data using the symmetric key for decryption.

b) What difference do you see between one algorithm (e.g. AES) and another algorithm of your choice?
Answer:
The difference is in cryptographic strength and performance.

2 CryptDB

I could not install CryptoDB from GitHab, so I used the docker container *mycrypt/cryptdb*.

Work with CryptoDB

1. Start cryptdb

```
$ sudo docker run -d -P --name cdb mycrypt/cryptdb
$ sudo docker exec -it cdb bash
/opt/cryptdb# cryptdb.sh start
```

2. Connect to CryptDB: (where root/letmein are username/password)

```
/opt/cryptdb# mysql -u root -pletmein -h 127.0.0.1 -P 3307
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql>
```

3. Perform simple queries:

```
mysql> CREATE DATABASE ASlab;
Query OK, 1 row affected (0.20 sec)

mysql> USE ASlab;
Database changed

mysql> create table t (name text, age integer);
Query OK, 0 rows affected (0.29 sec)

mysql> insert into t values ('alice', 19), ('bob', 20), ('chris', 21);
Query OK, 3 rows affected (0.13 sec)
```

```
mysql> select * from t;
+-------+------+
| name  | age  |
+-------+------+
| alice | 19   |
| bob   | 20   |
| chris | 21   |
+-------+------+
3 rows in set (0.06 sec)

mysql> select * from t where age = 19;
+-------+------+
| name  | age  |
+-------+------+
| alice | 19   |
+-------+------+
1 row in set (0.45 sec)

mysql> select sum(greatest(age,20)) from t;
+-----------------------+
| sum(greatest(age,20)) |
+-----------------------+
| 61                    |
+-----------------------+
1 row in set (0.58 sec)
```

4. Check that *CryptDB* works: (connect to *mysql* server)

```
/opt/cryptdb# mysql -u root -pletmein

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ASlab              |
| cryptdb_udf        |
| mysql              |
| performance_schema |
| remote_db          |
+--------------------+
6 rows in set (0.00 sec)

mysql> USE ASlab;
Database changed

mysql> SHOW TABLES;
```

```
+-------------------+
| Tables_in_ASlab   |
+-------------------+
| table_OYJBAAOFRU |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM table_OYJBAAOFRU;
+---------------------------------------------+--------------------+----┘
↪  ------------------+----------------------+-------------------+--------------┘
↪  ---------------------------------------------------------------------------┘
↪  ---------------------------------------------------------------------------┘
↪  ---------------------------------------------------------------------------┘
↪  -------------------+-----------------------+
| XSSBQRVYPLoEq                                 | ADLLFLNTSGoOrder     |
↪  cdb_saltTLOJGAQMDX   | LJDFMVZKUOoEq        | XWUIKBVXWNoOrder |
↪  EMAYSWYAKMoADD
↪
↪
↪                                 | cdb_saltPLSGWAIMML   |
+---------------------------------------------+--------------------+----┘
↪  ------------------+----------------------+-------------------+--------------┘
↪  ---------------------------------------------------------------------------┘
↪  ---------------------------------------------------------------------------┘
↪  ---------------------------------------------------------------------------┘
↪  -------------------+-----------------------+
| .LpaH /
                  2n[X |  9578623788662601074 | 1558464430757146160 |
                   ↪  840796299609358283 |      93623810461 | OF<7Zv-C6${*.^|l}┘
                   ↪          kjKp.WLM'ClZ]f]<$7s#PBxu`        ,_oWke}Ih74pi>
                   ↪  )*m4S[FF#ZmO<&Pv
aL;_!H|-/:FH
5M[`SzK |    68642374884255229 |
| 5t.OL6hARTLj>*i{,Idar | 16757368580540517162 |  6609833165821121230 |
↪  2557961935934523635 |      93853996078 | [LH@'.9 &ri*WIe)gz,x;\ @SF[TZ<
hT*9gC`>W%;^[w2XvAzR^\Q5?zAGeSzPQw lJAO{h"tR1PnW!;,.4&.5|
↪  1818332244855206021 |
| ?%G&oo/S-Gn:]|swJOa$3| 14968158576145523612 |  7115274902211751418 |
↪  17956853480120987546 |      100168944519 | |[ "OZQhFS8*;Uq'q 4Ej
+---------------------------------------------+--------------------+----┘
↪  ------------------+----------------------+-------------------+--------------┘
↪  ---------------------------------------------------------------------------┘
↪  ---------------------------------------------------------------------------┘
↪  ---------------------------------------------------------------------------┘
↪  -------------------+-----------------------+
3 rows in set (0.00 sec)
```

Unsupported queries in CryptoDB:

- More complex operators (*eg*, trigonometry)
- Operations that require combining encryption schemes (*eg*, $T1.a + T1.b > T2.c$)
- CryptDB can handle only four out of 22 TPC-H queries
- Order comparison
- Transactionally inserting into both a table and its corresponding ciphertext file(s).
- Text pattern matching with two or more patterns, such as c LIKE
- Both computation and comparison on the same column
- Queries on certain sensitive fields that perform string manipulation (e.g., substring and lowercase conversions) or date manipulation (e.g., obtaining the day, month, or year of an encrypted date).

The example of an UNIMPLEMENTED EXCEPTION:

```
mysql> SELECT name FROM t WHERE name LIKE '%a';
ERROR 1105 (07000): Error: Bad Query: [SELECT name FROM t WHERE name LIKE '%a']
Error Data: open_normal_and_derived_tables
FILE: main/rewrite_main.cc
LINE: 1380
```

Questions

a) Would you use this in a production environment?
Answer:
Probably not, because this system still needs improvement. But I found it very convenient when working with a database. In addition, it has very high performance despite the fact that it encrypts the data. The CryptoDB approach is applied in such companies as Google and Microsoft.

b) What are the concerns you would have if you do so?
Answer:
Problems would arise if you need to execute queries that are not supported by CryptoDB. And also it is necessary to increase the protection from unauthorized access to the CryptBD Proxy server.

3 Performance

Compare the performance of CryptDB with MS SQL Server 2017

Data was generated on the *www.convertcsv.com* resource. Number of records: **10 000**

DBMS	INSERT	SELECT
CryptDB	5 min 23 sec	4.30 sec
MS SQL Server 2017	2 min 02 sec	10 sec