

Lab: Data Encryption

- This is worth 2 points.
- The due date is tomorrow midnight.
- Use the following naming convention: homework, underscore, last name, first initial, and extension (e.g., Lab_Encrypt_Img.docx).

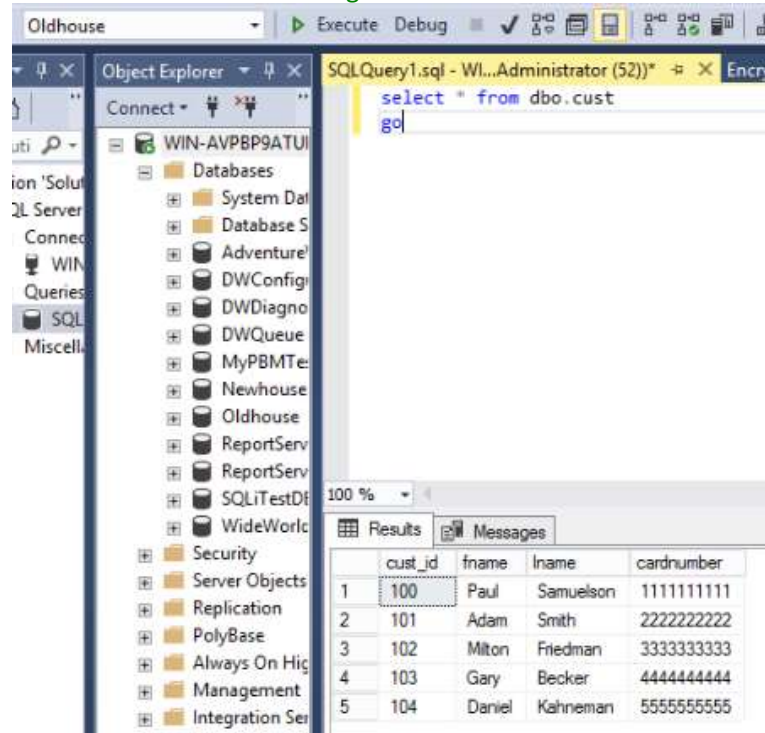
1. Preparation

First, if your SQL Server does not have Oldhouse database, create it using this script: **Oldhouse-Table-Create (Lab).sql**.

Next, perform the lab using this script: **Encryption-Cert (Lab).sql**.

2. Deliverables

```
-- Display the original table
select * from dbo.cust
go
/* Task #1: Show the original table in a screen shot. */
```



```
-- Display the encrypted table
select * from dbo.cust_encrypt
go
/* Task #2: Show the encrypted table in a screen shot. Also, explain why we need to
change the data type for encryption. */
```

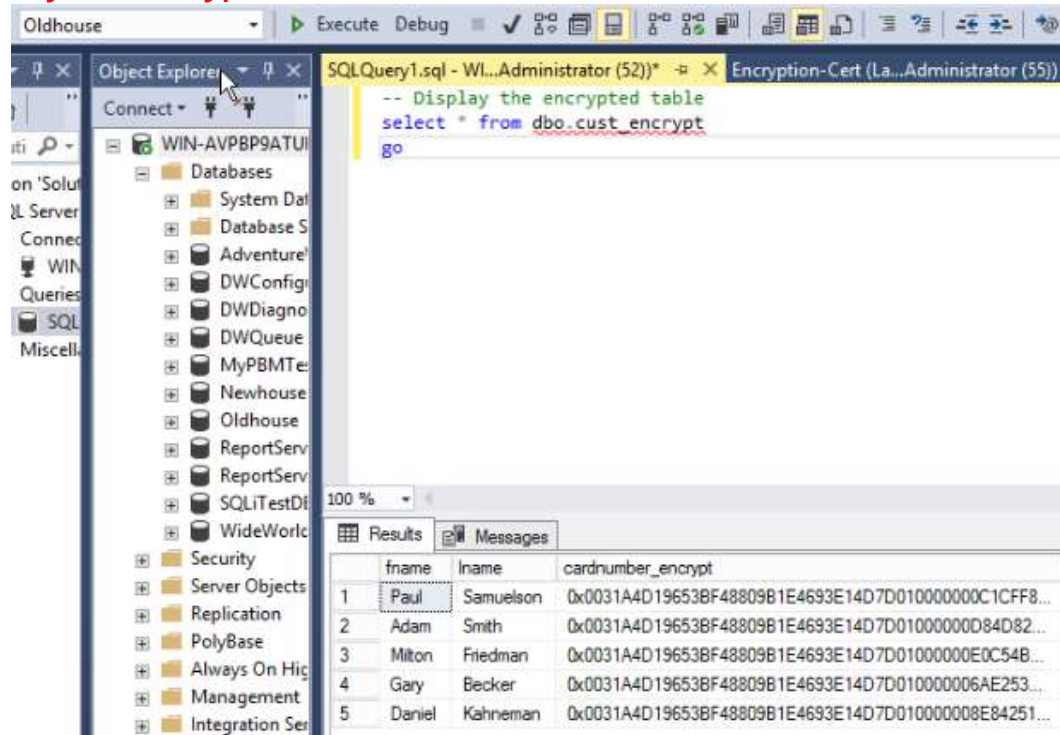
We need to convert it to varbinary because the EncryptByPassphrase returns that data type. The variable that stores the card number must be the same type.

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure, including the 'Oldhouse' database. The SQL Query window in the center shows the query: `select * from dbo.cust_encrypt`. The Results pane at the bottom displays the output of the query, showing five rows of data with columns: `fname`, `lname`, and `cardnumber_encrypt`. The data is as follows:

	fname	lname	cardnumber_encrypt
1	Paul	Samuelson	0x0100000072404DA165D57187CEF0220C9E2533709A6B556...
2	Adam	Smith	0x01000000031D52F7524A06B3FD3551A7D9FE9437DEC6E4...
3	Milton	Friedman	0x01000000608556E437F5CA50BD6D3F0538EED533170AB2...
4	Gary	Becker	0x01000000E970D752A22645A5BE6798441F875A15B8D5536...
5	Daniel	Kahneman	0x01000000756273815DC3391CF551DBF5E04FB11B4924485...

```
-- Display the encrypted table
select * from dbo.cust_encrypt
go
/* Task #3: Show the encrypted table in a screen shot. Also, explain the encryption
process after Task #2. */
```

I created a certificate which acted as the base for a symmetrical key. Then I created the symmetric key with the BillingCert certificate. This acted as the base I encrypted by. Finally I used the symmetric key to encrypt the data and stored it in the table.



The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure, including the 'Oldhouse' database. The SQL Query window on the right contains the following SQL code:

```
-- Display the encrypted table
select * from dbo.cust_encrypt
go
```

The Results pane at the bottom displays the output of the query as a table with three columns: 'fname', 'lname', and 'cardnumber_encrypt'. The table contains five rows of data, representing encrypted customer records.

	fname	lname	cardnumber_encrypt
1	Paul	Samuelson	0x0031A4D19653BF48809B1E4693E14D7D01000000C1CFF8...
2	Adam	Smith	0x0031A4D19653BF48809B1E4693E14D7D01000000D84D82...
3	Milton	Friedman	0x0031A4D19653BF48809B1E4693E14D7D01000000E0C54B...
4	Gary	Becker	0x0031A4D19653BF48809B1E4693E14D7D010000006AE253...
5	Daniel	Kahneman	0x0031A4D19653BF48809B1E4693E14D7D010000008E84251...

```
-- Display the decrypted table
select fname,
       lname,
       cardnumber = convert(nvarchar(25), DecryptByKey(cardnumber_encrypt))
from dbo.cust_encrypt
go
/* Task #4: Show the encrypted table in a screen shot. Also, explain the decryption
process after Task #3. */
/* Did you get the original data back? If not, what's wrong? */
/* Hint: Check out the current data type of cardnumber with the original one */
```

I used the Open Symmetric Key with the certificate we created in a previous step. This allowed an authorized user to access the data. I used DecryptByKey to decrypt the table using the Billing Cert key. Lastly, I output the data as a nvarchar with the convert function.

