How to encrypt data in SQL Server

Author: Eric Rouach, Madeira Data Solutions.

Date: 08/05/2021

The attached scripts demonstrate two common use-cases of sensitive-data encryption:

**Use-Case 1 – encrypt and keep the data decryptable**

Using as an example the AdventureWorks2014 database, the first script describes the process of encrypting the “CardNumber” column from the Sales.CreditCard table while keeping the data decryptable.

Our pre-requisite is the creation of a Master Key, a Certificate and a Symmetric Key.

Once having those created, we may proceed to the addition of a new column called “CardNumberEnc” (where the suffix “Enc” stands for “Encrypted”). This column has a VARBINARY(250) Data Type and is nullable.

We can then populate the newly created column with an encryption of the existing data by following these steps:

-Open the symmetric key

-Update the table and set the new column values using the “EncryptByKey” function.

Optionally, we may use an authenticator for an additional security layer (e.g. the ID column) using the syntax as shown in the following screenshot:



One the new column is populated, follow the following steps:

-Make the new column non-nullable

-Drop the old column

-Rename the new column to the old one’s name

-Close the symmetric key

-Create a nonclustered index on the new column

To decrypt the data, use the DecryptByKey function as shown is the script.

In case you have used an authenticator, use the following syntax:



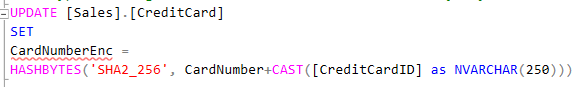
**Use-Case 2 – encrypt and make the data non-decryptable**

In this case, we do not need to create keys and/or certificates.

In a similar way as that of the first case, we will add a new column. The new column population process is the main difference here: we will use the HASHBYTES function as shown below:



You may use a salt as an extra security layer using the following syntax instead:



\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*