[Column Encryption in SQL Server 2008 with Symmetric Keys](http://dpatrickcaldwell.blogspot.ca/2010/10/column-encryption-in-sql-server-2008.html)

Most of the time when I write blog posts, I do it to share ideas with my fellow developers. Sometimes I do it just so I can have a place to reference when I forget the syntax for something. This is one of those reference posts.  
  
Recently I've been charged to column level encrypt some personally identifiable information. The present post is not intended to discuss the merits of column level encryption; rather, as I said it is to put a few code snippets up so that I can reference them later. If you should find yourself in a column level encryption predicament in a SQL Server 2008 environment, you may find these useful as well.

First thing's first. Get the database ready for column level encryption by creating a master key:

1. --if there is no master key create one
2. **IF** NOT EXISTS
3. (
4. **SELECT** \*
5. **FROM** sys.symmetric\_keys
6. **WHERE** symmetric\_key\_id = 101
7. )
8. **CREATE** **MASTER** **KEY** **ENCRYPTION** **BY**
9. **PASSWORD** = 'This is where you would put a really long key for creating a symmetric key.'
10. **GO**

Now, you'll need a certificate or a set of certificates with which you will encrypt your symmetric key or keys:

1. -- if the certificate doesn't, exist create it now
2. **IF** NOT EXISTS
3. (
4. **SELECT** \*
5. **FROM** sys.certificates
6. **WHERE** name = 'PrivateDataCertificate'
7. )
8. **CREATE** **CERTIFICATE** PrivateDataCertificate
9. **WITH** **SUBJECT** = 'For encrypting private data';
10. **GO**

Once you have your certificates, you can create your key or keys:

1. -- if the key doesn't exist, create it too
2. **IF** NOT EXISTS
3. (
4. **SELECT** \*
5. **FROM** sys.symmetric\_keys
6. **WHERE** name = 'PrivateDataKey'
7. )
8. **CREATE** **SYMMETRIC** **KEY** PrivateDataKey
9. **WITH** **ALGORITHM** = **AES\_256**
10. **ENCRYPTION** **BY** **CERTIFICATE** PrivateDataCertificate;
11. **GO**

Before you can use your symmetric key, you have to open it. I recommend that you get in the habit of closing it when you're finished with it. The symmetric key remains open for the life of the session. Let's say that you have a stored procedure in which you open the symmetric key to decrypt some private data which your stored procedure uses internally. Someone who has access to the stored procedure can run it and then will have the key opened for use in decrypting private data. My point, close the key before you leave the procedure. Here's how you open and close keys.

1. -- open the symmetric key with which to encrypt the data.
2. **OPEN** **SYMMETRIC** **KEY** PrivateDataKey
3. **DECRYPTION** **BY** **CERTIFICATE** PrivateDataCertificate;
5. -- close the symmetric key
6. **CLOSE** **SYMMETRIC** **KEY** PrivateDataKey;

Here's a little test script I wrote to demonstrate a few points. First, the syntax for encrypting and decrypting. Second, the fact that the the cipher text changes each time you do the encryption. This prevents a plain text attack.

1. -- open the symmetric key with which to encrypt the data.
2. **OPEN** **SYMMETRIC** **KEY** PrivateDataKey
3. **DECRYPTION** **BY** **CERTIFICATE** PrivateDataCertificate;
5. -- somewhere to put the data
6. **DECLARE** @TestEncryption **TABLE**
7. (
8. PlainText **VARCHAR**(100),
9. Cipher1 **VARBINARY**(100),
10. Cipher2 **VARBINARY**(100)
11. );
13. -- some test data
14. **INSERT** **INTO** @TestEncryption (PlainText)
15. **SELECT** 'Boogers'
16. **UNION** ALL
17. **SELECT** 'Foobar'
18. **UNION** ALL
19. **SELECT** '457-55-5462'; -- ignoranus
21. -- encrypt twice
22. **UPDATE** @TestEncryption
23. **SET**
24. Cipher1 = ENCRYPTBYKEY(KEY\_GUID('PrivateDataKey'), PlainText),
25. Cipher2 = ENCRYPTBYKEY(KEY\_GUID('PrivateDataKey'), PlainText);
27. -- decrypt and display results
28. **SELECT**
29. \*,
30. CiphersDiffer = CASE **WHEN** Cipher1 <> Cipher2 **THEN** 'TRUE' **ELSE** 'FALSE' **END**,
31. PlainText1 = CONVERT(**VARCHAR**, DECRYPTBYKEY(Cipher1)),
32. PlainText2 = CONVERT(**VARCHAR**, DECRYPTBYKEY(Cipher2))
33. **FROM** @TestEncryption;
35. -- close the symmetric key
36. **CLOSE** **SYMMETRIC** **KEY** PrivateDataKey;