MSSQLTips.com
brought to you by Edgewood Solutions

Your daily source for SQL Server Tips

SQL Server Encryption Symmetric vs. Asymmetric Keys

Written By: K. Brian Kelley -- 11/23/2009

Problem
I need to encrypt my data within SQL Server and I plan on using the built-in encryption functionality in SQL Server 2005 and 2008. However, I'm looking at symmetric and asymmetric key algorithms and while I see information saying to use symmetric keys, I don't understand why. What's the difference between the two and why is a symmetric key algorithm preferred over the asymmetric key ones?

Solution
Both asymmetric and symmetric key algorithms are encryption algorithms. Both include mathematical operations to take a known piece of information, like a person's social security number or national ID, and render it effectively unusable, unless you have the secret to change that data back to its original form. We call any secrets "keys" and one of the main differences between the two types of algorithms are the number of secrets, or keys, involved.

Symmetric

In a symmetric key algorithm, there is but one key. That same key is used to encrypt the data and unencrypt, or decrypt, the data. If someone were to get possession of the key, that person could take anything you've encrypted, and decrypt it immediately.

Figure 1 shows a visual example of encrypting data using a symmetric key algorithm. Figure 2 shows an example of decrypting the data. Note that there is only one key in both cases.

Figure 1:

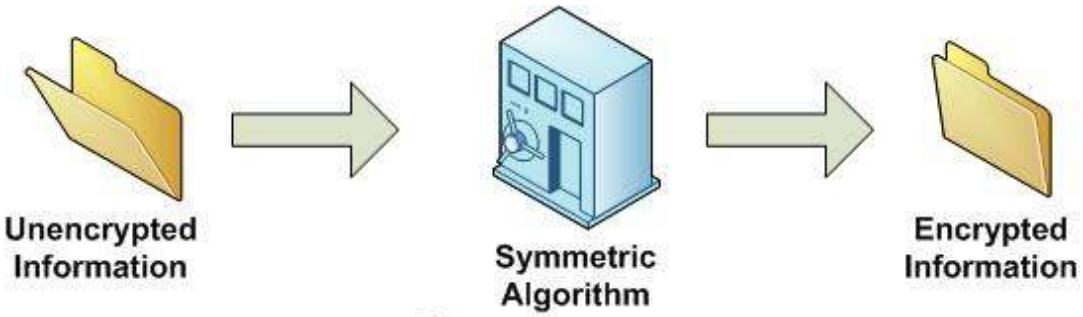
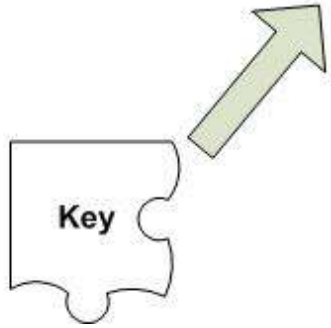
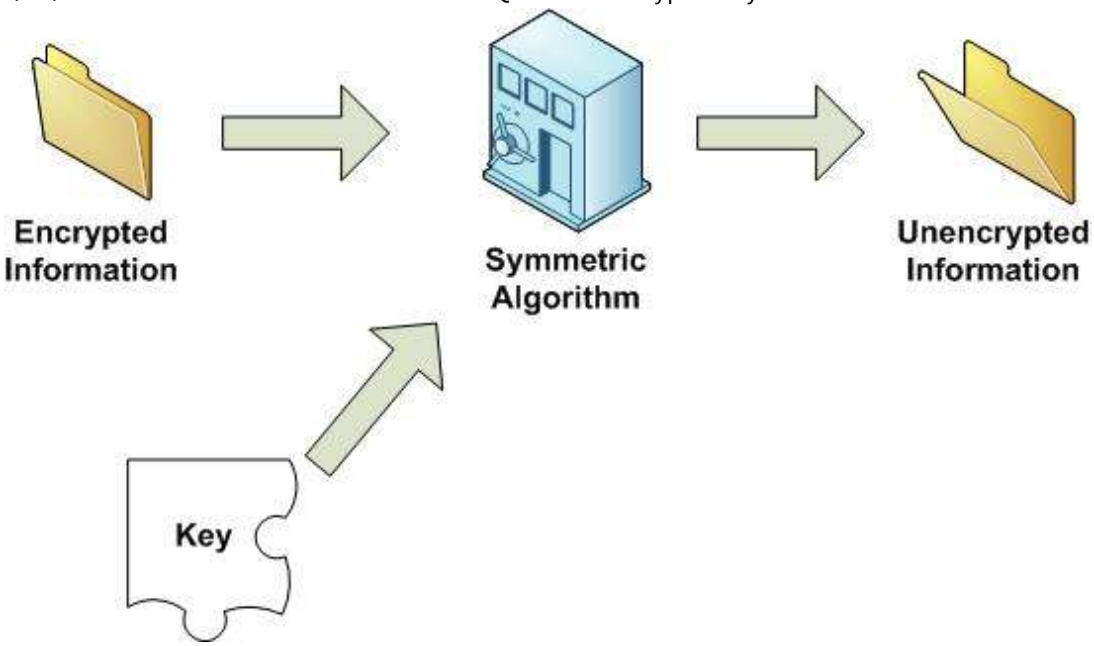


Figure 2:





Asymmetric

In an asymmetric encryption algorithm, usually called a public-private key algorithm, there are two keys. One can be made public. So if anyone wanted to send you something and encrypt it, they would just need your public key. The other key should be kept safe, and is called the private key. The way an asymmetric encryption algorithm works is that if someone encrypts data using your public key, only your private key can be used to decrypt the data. So as long as you keep the private key safe, no one can decrypt the data even if they have your public key.

Figures 3 and 4 show this idea:

Figure 3:

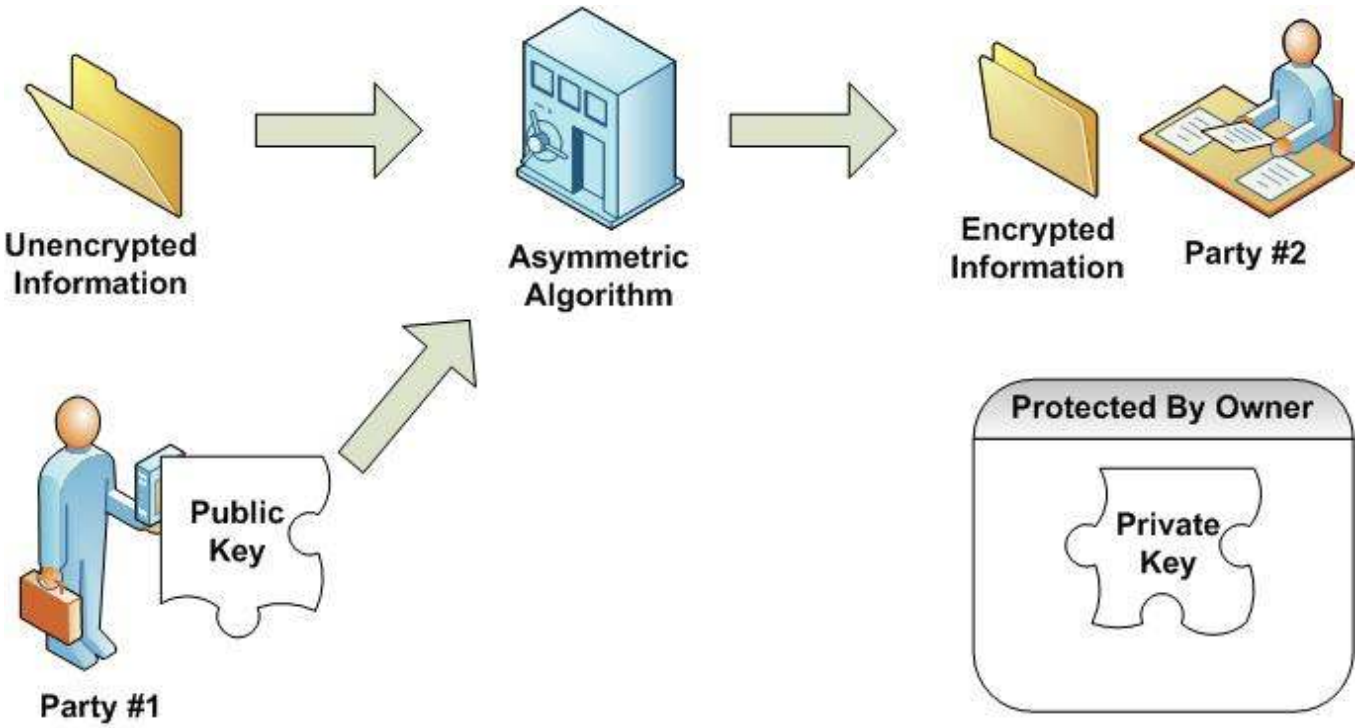
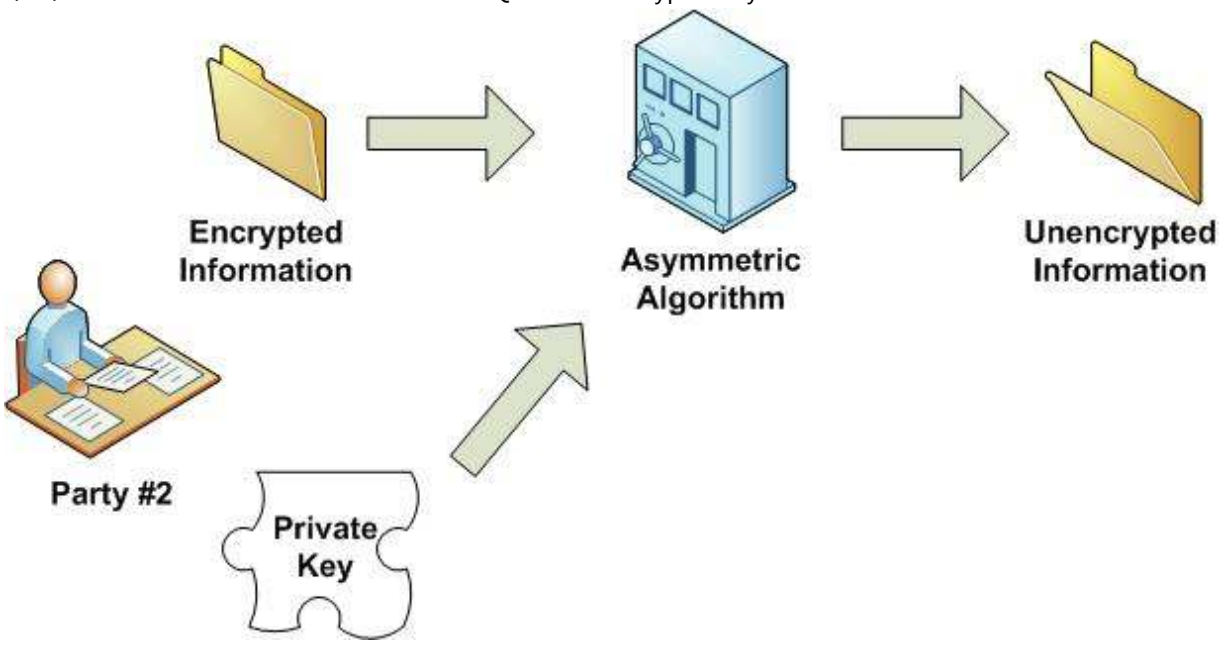


Figure 4:



With respect to SQL Server, it can do the key handling for us. So if we let it, SQL Server's built-in encryption functionality keeps track of all these details and for practical purposes, there is not any difference between symmetric and asymmetric keys. You can encrypt data and you can decrypt data and both seem to work fine.

Asymmetric or Symmetric?

So why then is there a recommendation to use symmetric keys to encrypt data? Quite simply, performance. Symmetric key algorithms tend to be mathematically simpler, and as a result, faster. The difference in speed can be significant even into the 100x faster range. Therefore, symmetric key algorithms are the way to go when encrypting data.

We can see this difference quite clearly with a simple example. We can set up two tables and create two keys. One set will be for an asymmetric key algorithm. The other will be for a symmetric key algorithm. And we can run through a number of rows of data and determine how much time it takes between the two algorithms. What we should see is that the symmetric key encryption is performed a noticeable amount faster.

First, let's do the setup:

```
USE MSSQLTips;
GO

/* Setup for Testing */
CREATE SYMMETRIC KEY TestSymmKey
WITH ALGORITHM = AES_256
ENCRYPTION BY PASSWORD = 'TestP4ssw0rd!';
GO

CREATE ASYMMETRIC KEY TestAsymmKey
WITH ALGORITHM = RSA_512
ENCRYPTION BY PASSWORD = 'TestP4ssw0rd!';
GO

CREATE TABLE dbo.SymmKeyTest (
  EncryptedCol VARBINARY(256)
);
GO

CREATE TABLE dbo.AsymmKeyTest (
  EncryptedCol VARBINARY(256)
);
GO
```

Next, the symmetric key test:

```
/* Symmetric Key Test */
```

```
DECLARE @StartTime DATETIME;
DECLARE @EndTime DATETIME;
DECLARE @KeyGUID UNIQUEIDENTIFIER;

SET @KeyGUID = KEY_GUID('TestSymmKey');
SET @StartTime = GETDATE();

OPEN SYMMETRIC KEY TestSymmKey DECRYPTION BY PASSWORD = 'TestP4ssw0rd!';

INSERT INTO dbo.SymmKeyTest (EncryptedCol)
SELECT TOP 5000 ENCRYPTBYKEY(@KeyGUID, 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz')
FROM master.sys.columns c1 CROSS JOIN master.sys.columns c2;

SELECT TOP 5000 CONVERT(VARCHAR(52), DECRYPTBYKEY(EncryptedCol))
FROM dbo.SymmKeyTest;

SET @EndTime = GETDATE();

PRINT 'Symmetric Key Time Difference (ms): ' + CONVERT(CHAR, DATEDIFF(ms, @StartTime, @EndTime));
GO
```

Then, the asymmetric key test:

```
/* Asymmetric Key Test */
DECLARE @StartTime DATETIME;
DECLARE @EndTime DATETIME;
DECLARE @AsymID INT;

SET @AsymID = ASYMKEY_ID('TestAsymmKey');
SET @StartTime = GETDATE();

INSERT INTO dbo.AsymmKeyTest (EncryptedCol)
SELECT TOP 5000 ENCRYPTBYASYMKEY(@AsymID, 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz')
FROM master.sys.columns c1 CROSS JOIN master.sys.columns c2;

SELECT TOP 5000 CONVERT(CHAR(52), DECRYPTBYASYMKEY(@AsymID, EncryptedCol, N'TestP4ssw0rd!'))
FROM dbo.AsymmKeyTest;

SET @EndTime = GETDATE();

PRINT 'Asymmetric Key Time Difference (ms): ' + CONVERT(VARCHAR, DATEDIFF(ms, @StartTime, @EndTime));
GO
```

If you've kept the time of each, even though we did a relatively simple example with a small number of rows (5,000), you should see a noticeable difference in times. This gets worse the more data we're encrypting and decrypting.

Figures 5 and 6 show the difference in times on one of my systems.

Figure 5: - Symmetric Test

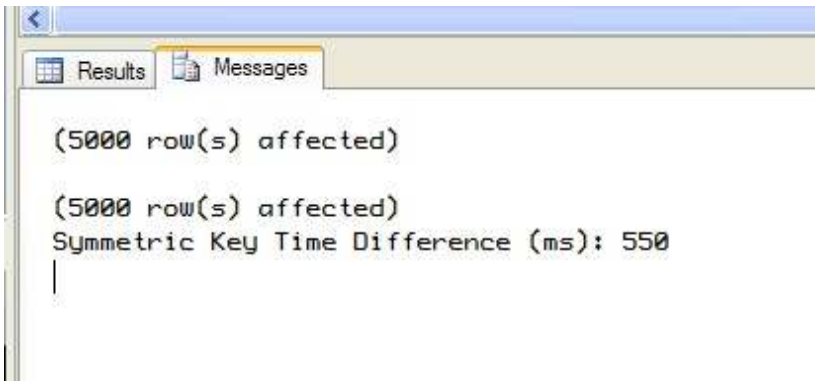
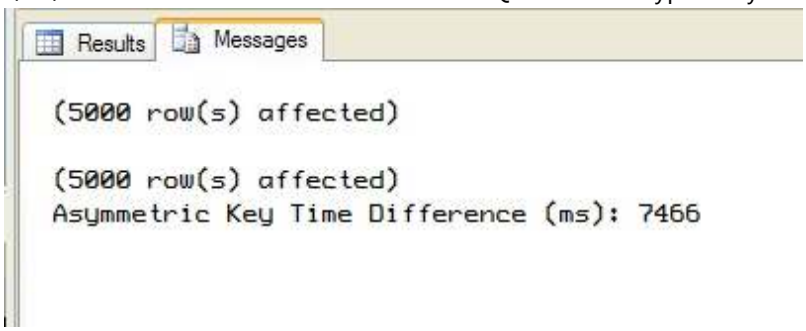


Figure 6: - Asymmetric Test



In my testing, I saw times averaging around 550 milliseconds for the symmetric key pass. And I saw times around 7500 milliseconds for the asymmetric key pass. So even in this simple example of 5000 rows of 52 characters, we can already see a sizeable difference of about 7 full seconds. Your results will vary based on your equipment and existing load at the time. But you should see a comparable difference where the symmetric key pass is better than 10x faster than the asymmetric key one. And that's why symmetric keys are preferred to encrypt data over asymmetric keys. It's more about performance than anything else.

Next Steps

- Do some testing in your environment to see which approach is better
- Read more tips about encryption
  - SQL Server Encryption Key Management between Development, Test and Production Environments
  - Natively Encrypting Social Security Numbers in SQL Server 2005
  - SQL Server 2005 Encryption - Certificates 101
  - Managing SQL Server 2005 Master Keys for Encryption