.net column

# Cryptografie in .NET

Wanneer gegevens uitgewisseld worden tussen applicaties, is het van belang dat gevoelige informatie niet in handen van onbevoegden valt. Een oplossing daarvoor is het versleutelen (encrypten) van de gegevens. Deze encryptie is natuurlijk met name van belang wanneer de gegevens uitgewisseld worden over een onbeveiligde verbinding, omdat de kans dat ze in verkeerde handen terechtkomen dan veel groter is. Vandaar dat er een speciale API bestaat, de CryptoAPI, die functies bevat die deze encryptie kunnen uitvoeren. In .NET is er een namespace, System.Security.Cryptography, waarin classes zitten die een groot deel van deze API aan .NET-applicaties beschikbaar stellen.

Wanneer we de classes in deze namespace onder de loep nemen, valt in eerste instantie op dat de classes die encryptie-algoritmes implementeren, in twee groepen uiteenvallen: de symmetrische algoritmes en de asymmetrische algoritmes. Symmetrische algoritmes kenmerken zich door het feit dat encryptie en decryptie met dezelfde sleutel plaatsvindt. Nadeel van deze groep algoritmes is dus dat de zender en de ontvanger van een bericht beiden over dezelfde sleutel moeten beschikken. Voordeel is dat encryptie door middel van een symmetrische algoritme veel sneller is dan encryptie via een asymmetrisch algoritme. Deze classes worden dan ook gebruikt voor het encrypten van grote hoeveelheden data. In .NET gebruik je deze classes door een CryptoStream te creëren op een bestaande Stream, zoals in het codevoorbeeld linksonder op deze pagina te zien is.

```
using System.IO;
using System.Security.
Cryptography;
FileStream fs = new
FileStream("test.
crp",FileMode.Create);
TripleDESCryptoServiceProvi
der tdcsp = new
    TripleDESCryptoService
Provider();
CryptoStream cs = new
CryptoStream(fs, tdcsp.
CreateEncryptor(),CryptoStr
eamMode.Write);
StreamWriter sw = new
StreamWriter(cs);
sw.WriteLine("Dit wordt
versleuteld");
sw.Close();
```

In dit voorbeeld wordt eerst een gewone FileStream gecreëerd, vervolgens wordt een object geïnstantieerd van een class die een symmetrisch algoritme implementeert (TripleDES CryptoServiceProvider). Hierbij wordt automatisch een nieuwe sleutel (key) gegenereerd. Daarna wordt een CryptoStream gecreëerd, waarbij de bestaande FileStream en een Encryptor als parameter meegegeven worden. De betreffende Encryptor is gecreëerd door het TripleDESCryptoServiceProvider-object. Op basis van deze CryptoStream wordt vervolgens een StreamWriter geïnstantieerd, die gebruikt kan worden om tekst weg te schrijven. Alles wat via deze StreamWriter weggeschreven wordt, wordt automatisch geëncrypt.

Om de weggeschreven (geëncrypte) gegevens weer in te kunnen lezen, kunnen we vergelijkbare code gebruiken als bij het wegschrijven, behalve dat er nu een Decryptor gecreëerd moet worden. Bij het creëren van deze Decryptor moet dan wel dezelfde sleutel (Key en IV) meegegeven worden, anders is het niet mogelijk dit bericht te decrypten. Deze sleutelinformatie kan uit het bestaande TripleDESCryptoServiceProvider-object verkregen worden via de properties Key en IV.

De asymmetrische algoritmes maken gebruik van een key-paar (Public Key en Private Key). Het idee hierachter is dat iedereen zo'n key-paar heeft en dat de Public Key openbaar verspreid kan worden, maar dat de Private Key strikt geheim gehouden wordt.

Wanneer iemand iets versleuteld naar een ander wil sturen, dan moet hij eerst de Public Key van die ander in zijn bezit krijgen. Omdat deze key openbaar is, kan deze gewoon via een website of via mail uitgewisseld worden. Vervolgens encrypt hij de boodschap met deze Public Key. Nu kan alleen degene die de Private Key heeft (de ontvanger van het bericht dus), dit bericht lezen. Omdat deze algoritmes niet geschikt zijn voor het encrypten van grote hoeveelheden data, werken ze niet, zoals de symmetrische algoritmes, op Streams, maar op een array van bytes. De meest bekende asymmetrische algoritmes zijn DSA en RSA.

Meestal worden de asymmetrische algoritmes gebruikt om een sleutel uit te wisselen, waarmee vervolgens het te sturen bericht via een symmetrisch algoritme versleuteld wordt. Zo worden de voordelen van beide soorten algoritmes gecombineerd: door het gebruik van het asymmetrische algoritme wordt voorkomen dat op voorhand een sleutel bij beide partijen bekend moet zijn en door het gebruik van het symmetrische algoritme wordt de nodige efficiëntie bij het versleutelen verkregen.

*Gert Jan Timmerman, hoofd kenniscentrum bij Info Support*