

# BCP – Ejemplos criptografía

Proyectos de encriptación y desencriptación de configuración

Cliente:

BCP

Preparado por:

Gabriel Lopardo – Developer

4-Jun-18

## Table of Contents

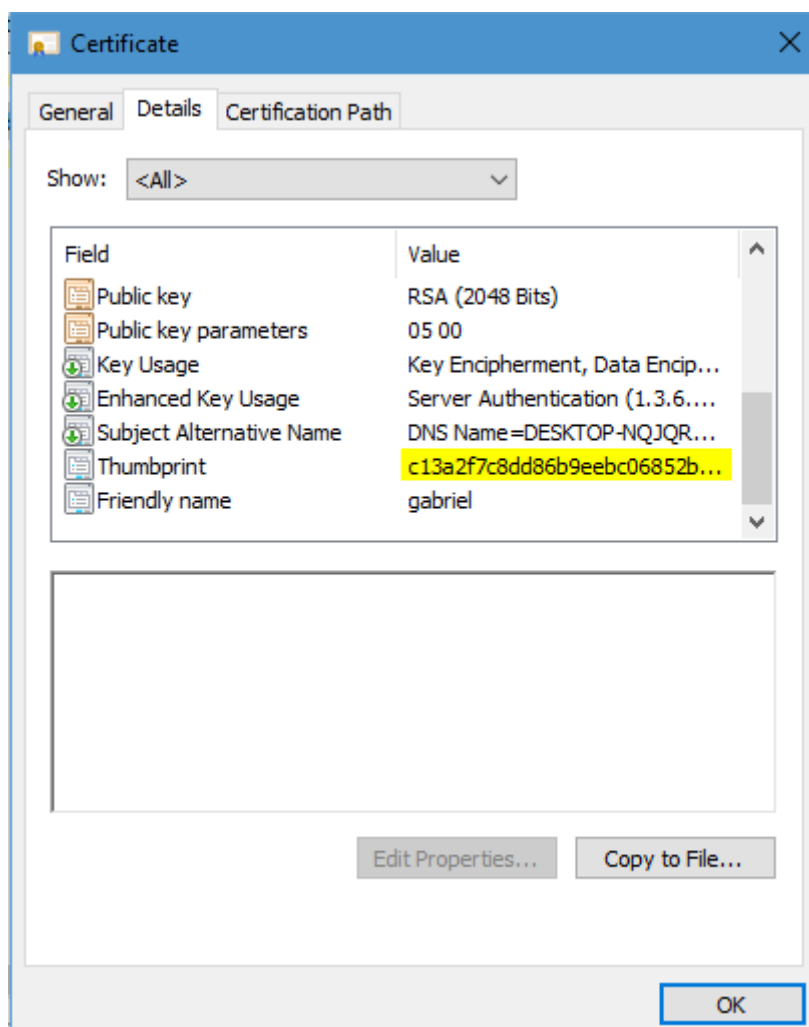
Encriptación y desencriptación de archivos de configuración .json con .ASP NET Core .....	3
1. Encriptación .....	3
a. ¿Qué hacer con la información encriptada? .....	4
2. Desencriptación.....	5

# Encriptación y desencriptación de archivos de configuración .json con .ASP NET Core

En esta solución se pretende ilustrar un caso de encriptación de un archivo de configuración para una aplicación web en .ASP NET Core 2.0 con el agregado de un ConfigurationProvider custom para separar la configuración con datos sensibles en un archivo aparte que no se publica en producción

## 1. Encriptación

Se debe crear un server certificate (cualquier tipo de certificado) en IIS y copiar su thumbprint, la usaremos para identificar el certificado y posteriormente acceder a sus claves pública y privada.



Para la encriptación se utiliza una aplicación de consola que va a leer dos parámetros (de a uno por vez) para devolverlos en formato .txt con una estructura similar a la de un archivo de configuración .json para facilitar trasladar allí los datos.

El primer parámetro es el **id cliente** que es meramente informativo, ya que esta aplicación permite hacer un ingreso masivo de clientes.

El ingreso de un idCliente vacío finaliza la carga masiva.

El segundo parámetro es la información que queremos encriptar, que en este caso se ingresan 3 **connection strings**.

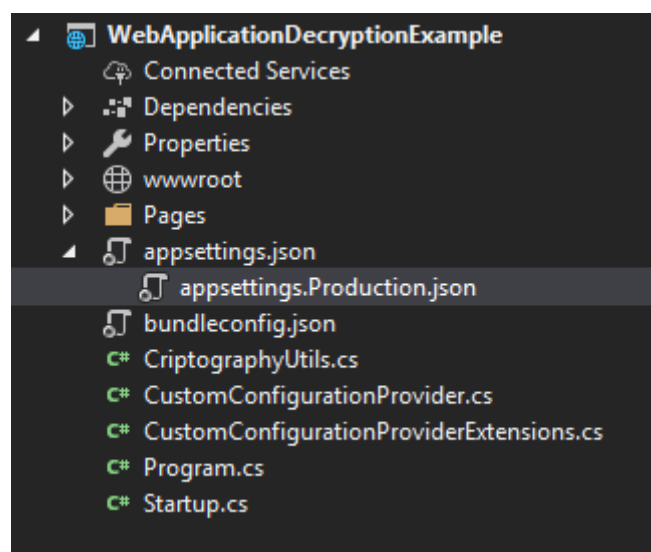
El resultado será un archivo con el siguiente formato:

```
"idCliente_1": "informacionEncriptada_1",
"idCliente_2": "informacionEncriptada_2",
...
"idCliente_n": "informaciónEncriptada_n"
```

Donde informacionEncriptada se representa como un string expresado en base 64, de la data ingresada como segundo parámetro, encriptada utilizando el algoritmo RSA y la clave pública del certificado creado en el primer paso.

## a. ¿Qué hacer con la información encriptada?

Primero se debe crear un archivo de configuración en la solución con el nombre appsettings.ENVIRONMENT.json donde ENVIRONMENT va a ser el ambiente correspondiente (development, staging, production) definido en las propiedades del proyecto en la variable de entorno ASPNETCORE\_ENVIRONMENT.



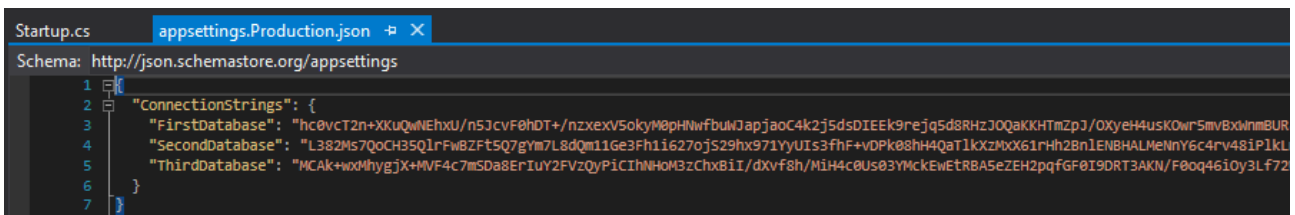
Una vez agregado el archivo (en este caso appsettings.production.json) el mismo se anidará a appsettings.json.

En el constructor de la clase Startup al instanciar un nuevo ConfigurationBuilder se deberá llamar al método custom (AddCustomProvider) que devuelve una instancia de CustomConfigurationBuilder y luego al método de extensión AddJsonFiles que se le pasa como parámetro el nombre del archivo de configuración que vamos a agregar.

```
public Startup(IHostingEnvironment env)
{
    var builder = new ConfigurationBuilder()
        .SetBasePath(env.ContentRootPath)
        .AddJsonFile("appsettings.json", true, true)
        .AddCustomProvider()
        .AddJsonFile($"appsettings.{env.EnvironmentName}.json", true, true);

    _configuration = builder.Build();
}
```

Luego podemos copiar la información que encriptamos con la aplicación (siguiendo el formato correspondiente).



```
Schema: http://json.schemastore.org/appsettings
{
  "ConnectionStrings": {
    "FirstDatabase": "hc0vCT2n+XKuQwNEhxU/n5JcvF0hDT+/nzxexV5okyM0pHNwfbuWJapJaoC4k2j5dsDIEEk9rejq5d8RH2JOQaKKHTmzpJ/OxyeH4usKOWr5mVbXwNmBUR",
    "SecondDatabase": "L382Ms7QoCH35QlrFw8ZFt5Q7gYm7L8dQm11Ge3Fh11627ojs29hx971YyUIS3fhF+vDPk08hH4QaTlkxzMxx61rHh2Bn1ENBHALMeNnY6c4rv48iPlkL",
    "ThirdDatabase": "MCAk+wo4hygjX+MVF4c7mSDa8ErIuY2FV2QyPiCIhW0M3zChxBiI/dXvf8h/MiH4c0Us03YMckEwEtRBA5eZEH2pqFGF0I9DRT3AKN/F0oq46i0y3Lf72"
  }
}
```

## 2. Desencriptación

Se sigue el proceso inverso, esta vez utilizando la clave privada del certificado mencionado anteriormente, se lee el archivo de configuración con la data encriptada, la convertimos en una cadena de bytes y se la pasamos como parámetro al método Decrypt() de la clase CriptographyUtils, quien también nos retorna una cadena de bytes que podremos pasar a un string si así lo deseáramos.

```
var decryptedConnStr1 = Encoding.UTF8.GetString(CriptographyUtils.Decrypt(Convert.FromBase64String(connStr1), _configuration["CertificateThumbPrint"]));
var decryptedConnStr2 = Encoding.UTF8.GetString(CriptographyUtils.Decrypt(Convert.FromBase64String(connStr2), _configuration["CertificateThumbPrint"]));
var decryptedConnStr3 = Encoding.UTF8.GetString(CriptographyUtils.Decrypt(Convert.FromBase64String(connStr3), _configuration["CertificateThumbPrint"]));
```