大樹下玩耍

OpenSSL.Net使用隨記（二）

> 前面已經把使用OpenSSL.Net環境準備好了，現在來調用幾個常用算法的實現

- **MD5，SHA1**

在這只需要注意下OpenSSL.Crypto.MessageDiges後面簽名算法會用到。

```
class Program_Hash
 {
     static  void Main( string [] args)
     {
         var ciphertext = MD5( " Md5加密。" , Encoding.UTF8);
         ciphertext = SHA1( " SHA1加密。" , Encoding.UTF8);
     }

    public  static  string MD5( string text, Encoding encodir
     {
         return HashDigest(text, encoding, MessageDigest.MD5
     }

     public  static  string SHA1( string text, Encoding encoc
     {
         return HashDigest(text, encoding, MessageDigest.SHA
     }

     private  static  string HashDigest( string text, Encodir
     {
         using (MessageDigestContext hashDigest = new Messag
         {
             byte [] hashBytes =encoding.GetBytes(text);
             byte [] signByte = hashDigest.Digest(hashBytes)
             return BitConverter.ToString(signByte).Replace
         }
     }
 }
```

- **AES**

1、在這裡SymmetricCrypt是之前項目封裝的System.Security.Cryptography.RijndaelManaged

2、Key與IV是有其規律的因此IV是可以去掉

3、在string與byte[]之間轉換時要注意編碼，什麼時候用Convert.FromBase64String，什麼時候用Encoding

導航

博客園
首頁
新隨筆
聯繫
訂閱
管理

公告

暱稱： Azeri
園齡： 10年3個月
粉絲： 4
關注： 130
+加關注

< 2021年3月 >

日	一	二	三	四	五	六
28	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

搜索

找找看
谷歌搜索

常用鏈接

我的隨筆
我的評論
我的參與
最新評論
我的標籤

我的標籤

OpenSSL.Net (4)
SFTP (1)

隨筆檔案

2018年9月(1)
2018年5月(2)
2018年4月(2)

閱讀排行榜

1. SFTP免密碼登錄踏坑經歷(4521)
2. OpenSSL.Net使用隨記(536)
3. OpenSSL.Net使用隨記（二）(331)
4. OpenSSL.Net使用隨記（四）(321)
5. OpenSSL.Net使用隨記（三）(304)

4、可以查看下Cipher封装，裡面有不同的算法

```
class Program_Symmetric
{
    static void Main(string[] args)
    {
        SymmetricCrypt symmetric = new SymmetricCrypt(CryptTy
        string key = "PGKQBXNCiuwKmlIUThSy1h+ZHMAN+HytbZny/Fe
            iv = "aqauVvyO7qvAbaDsdOeFsA==",
            text = "AES256加解密。";

        var ctext = symmetric.Encrypt(text, key, iv);

        var ctext2 = symmetric.Decrypt(ctext, key, iv);

        var ctext3 = Encrypt(ctext2, key, iv);

        Decrypt(ctext3, key, iv);
    }

    public static string Decrypt(string text, string key, str
    {
        byte[] keyBytes = Convert.FromBase64String(key);
        byte[] ivBytes = Convert.FromBase64String(iv);
        byte[] textBytes = Convert.FromBase64String(text);
        using (CipherContext cipher = new CipherContext(Ciphe
        {
            byte[] output = cipher.Decrypt(textBytes, keyByte
            var result = Encoding.UTF8.GetString(output);
            return result;
        }
    }

    public static string Encrypt(string text, string key, str
    {
        byte[] keyBytes = Convert.FromBase64String(key);
        byte[] ivBytes = Convert.FromBase64String(iv);
        byte[] textBytes = Encoding.UTF8.GetBytes(text);
        using (CipherContext cipher = new CipherContext(Ciphe
        {
            byte[] output = cipher.Encrypt(textBytes, keyByte
            var result = Convert.ToBase64String(output);
            return result;
        }
    }
}
```

- **RSA**

1、公钥与私钥可以用OpenSSL命令行随意生成

2、OpenSSL.Core.BIO简单理解是用于装载密钥的容器

3、注意OpenSSL.Crypto.RSA的静态方法

4、OpenSSL.Crypto.CryptoKey用于装载BIO能把密钥转换成具体的算法对象，这个类的作用很大，涉及到签名验签都会用到

```
class Program_RSA
    {
        static void Main(string[] args)
        {
            string privateKey = "", publicKey = "", text = "RSA-
            int padding = 1;
            Encoding encoding = Encoding.UTF8;
            using (RSA rsa = new RSA())
            {
                rsa.GenerateKeys(1024, BigNumber.One, null, null
                privateKey = rsa.PrivateKeyAsPEM;
                publicKey = rsa.PublicKeyAsPEM;
            }

            ctext = PrivateEncrypt(privateKey, text, encoding, p
            text = PublicDecrypt(publicKey, ctext, encoding, pad

            ctext = PublicEncrypt(publicKey, text, encoding, pad
            text = PrivateDecrypt(privateKey, ctext, encoding, p

            var signText = Sign(privateKey, text, encoding);
            var signTag = Verify(publicKey, text, signText, enco

        }

        /// <summary>
        /// 私钥解密
        /// </summary>
        public static string PrivateDecrypt(string privateKey, s
        {
            byte[] textBytes = Convert.FromBase64String(text);
            using (BIO bio = new BIO(privateKey))
            {
                using (RSA rsa = RSA.FromPrivateKey(bio))
                {
                    textBytes = rsa.PrivateDecrypt(textBytes, (P
                }
            }
            return encoding.GetString(textBytes);
        }

        /// <summary>
        /// 私钥加密
        /// </summary>
        public static string PrivateEncrypt(string privateKey, s
        {
            byte[] textBytes = encoding.GetBytes(text);
            using (BIO bio = new BIO(privateKey))
            {
                using (RSA rsa = RSA.FromPrivateKey(bio))
                {
                    textBytes = rsa.PrivateEncrypt(textBytes, (P
                }
            }
```

```
55              return Convert.ToBase64String(textBytes);
56          }
57
58          /// <summary>
59          /// 公钥解密
60          /// </summary>
61          public static string PublicDecrypt(string publicKey, str
62          {
63              byte[] textBytes = Convert.FromBase64String(text);
64              using (BIO bio = new BIO(publicKey))
65              {
66                  using (RSA rsa = RSA.FromPublicKey(bio))
67                  {
68                      textBytes = rsa.PublicDecrypt(textBytes, (RS
69                  }
70              }
71              return encoding.GetString(textBytes);
72          }
73
74          /// <summary>
75          /// 公钥加密
76          /// </summary>
77          public static string PublicEncrypt(string publicKey, str
78          {
79              byte[] textBytes = encoding.GetBytes(text);
80              using (BIO bio = new BIO(publicKey))
81              {
82                  using (RSA rsa = RSA.FromPublicKey(bio))
83                  {
84                      textBytes = rsa.PublicEncrypt(textBytes, (RS
85                      rsa.Dispose();
86                  }
87                  bio.Dispose();
88              }
89              return Convert.ToBase64String(textBytes);
90          }
91
92          /// <summary>
93          /// 私钥签名
94          /// </summary>
95          public static string Sign(string privateKey, string text
96          {
97              using (BIO bio = new BIO(privateKey))
98              {
99                  using (CryptoKey cryptoKey = CryptoKey.FromPriva
100                 {
101                     using (MessageDigestContext sha256 = new Mes
102                     {
103                         byte[] msgByte = encoding.GetBytes(text)
104                         byte[] signByte = sha256.Sign(msgByte, c
105                         return Convert.ToBase64String(signByte);
106                     }
107                 }
108             }
109         }
110
111         /// <summary>
112         /// 公钥验签
```

```
        /// </summary>
        public static bool Verify(string publicKey, string text,
        {
            using (BIO bio = new BIO(publicKey))
            {
                using (CryptoKey cryptoKey = CryptoKey.FromPubli
                {
                    using (MessageDigestContext sha256 = new Mes
                    {
                        byte[] msgByte = encoding.GetBytes(text)
                        byte[] signByte = Convert.FromBase64Stri
                        return sha256.Verify(msgByte, signByte,
                    }
                }
            }
        }
    }
```

标签: OpenSSL.Net

« 上一篇： OpenSSL.Net使用随记
» 下一篇： OpenSSL.Net使用随记（三）

posted on 2018-04-30 16:48 Azeri 阅读(331) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部