

[首頁](#)[新聞](#)[博問](#)[專區](#)[閃存](#)[班級](#)[代碼改變世界](#)[註冊](#)[登錄](#)

## 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。

```
1  class Program_Hash
2  {
3      static void Main( string [] args)
4      {
5          var ciphertext = MD5( " Md5加密. ", Encoding.UTF8);
6          ciphertext = SHA1( " SHA1加密. ", Encoding.UTF8);
7      }
8
9      public static string MD5( string text, Encoding encoding)
10     {
11         return HashDigest(text, encoding, MessageDigest.MD5);
12     }
13
14     public static string SHA1( string text, Encoding encoding)
15     {
16         return HashDigest(text, encoding, MessageDigest.SHA1);
17     }
18
19     private static string HashDigest( string text, Encoding encoding)
20     {
21         using (MessageDigestContext hashDigest = new MessageDigestContext())
22         {
23             byte [] hashBytes = encoding.GetBytes(text);
24             byte [] signByte = hashDigest.Digest(hashBytes);
25             return BitConverter.ToString(signByte).Replace(" ", "");
26         }
27     }
28 }
```

#### • 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

  

#### 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封裝，裡面有不同的算法

```

1  class Program_Symmetric
2  {
3      static void Main(string[] args)
4      {
5          SymmetricCrypt symmetric = new SymmetricCrypt(CryptT
6          string key = "PGKQBXCiuwKmlIUThSy1h+ZHMAN+HytbZny/F
7              iv = "aqauVvy07qvAbaDsdOeFsA==",
8              text = "AES256加解密。";
9
10         var ctext = symmetric.Encrypt(text, key, iv);
11
12         var ctext2 = symmetric.Decrypt(ctext, key, iv);
13
14         var ctext3 = Encrypt(ctext2, key, iv);
15
16         Decrypt(ctext3, key, iv);
17     }
18
19     public static string Decrypt(string text, string key, sti
20     {
21         byte[] keyBytes = Convert.FromBase64String(key);
22         byte[] ivBytes = Convert.FromBase64String(iv);
23         byte[] textBytes = Convert.FromBase64String(text);
24         using (CipherContext cipher = new CipherContext(Ciphe
25         {
26             byte[] output = cipher.Decrypt(textBytes, keyByte
27             var result = Encoding.UTF8.GetString(output);
28             return result;
29         }
30     }
31
32     public static string Encrypt(string text, string key, sti
33     {
34         byte[] keyBytes = Convert.FromBase64String(key);
35         byte[] ivBytes = Convert.FromBase64String(iv);
36         byte[] textBytes = Encoding.UTF8.GetBytes(text);
37         using (CipherContext cipher = new CipherContext(Ciphe
38         {
39             byte[] output = cipher.Encrypt(textBytes, keyByte
40             var result = Convert.ToBase64String(output);
41             return result;
42         }
43     }
44 }

```

Powered by:

博客園

Copyright © 2021 Azeri

Powered by .NET 5.0 on Kubernetes

- 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

```

1 class Program_RSA
2 {
3     static void Main(string[] args)
4     {
5         string privateKey = "", publicKey = "", text = "RSA-
6         int padding = 1;
7         Encoding encoding = Encoding.UTF8;
8         using (RSA rsa = new RSA())
9         {
10             rsa.GenerateKeys(1024, BigInteger.One, null, null);
11             privateKey = rsa.PrivateKeyAsPEM;
12             publicKey = rsa.PublicKeyAsPEM;
13         }
14
15         ctext = PrivateEncrypt(privateKey, text, encoding, padding);
16         text = PublicDecrypt(publicKey, ctext, encoding, padding);
17
18         ctext = PublicEncrypt(publicKey, text, encoding, padding);
19         text = PrivateDecrypt(privateKey, ctext, encoding, padding);
20
21         var signText = Sign(privateKey, text, encoding);
22         var signTag = Verify(publicKey, text, signText, encoding);
23
24     }
25
26     /// <summary>
27     /// 私钥解密
28     /// </summary>
29     public static string PrivateDecrypt(string privateKey, string ctext)
30     {
31         byte[] textBytes = Convert.FromBase64String(ctext);
32         using (BIO bio = new BIO(privateKey))
33         {
34             using (RSA rsa = RSA.FromPrivateKey(bio))
35             {
36                 textBytes = rsa.PrivateDecrypt(textBytes, padding);
37             }
38         }
39         return encoding.GetString(textBytes);
40     }
41
42     /// <summary>
43     /// 私钥加密
44     /// </summary>
45     public static string PrivateEncrypt(string privateKey, string text)
46     {
47         byte[] textBytes = encoding.GetBytes(text);
48         using (BIO bio = new BIO(privateKey))
49         {
50             using (RSA rsa = RSA.FromPrivateKey(bio))
51             {
52                 textBytes = rsa.PrivateEncrypt(textBytes, padding);
53             }
54         }
55     }

```

```
55         return Convert.ToBase64String(textBytes);
56     }
57
58     /// <summary>
59     /// 公钥解密
60     /// </summary>
61     public static string PublicDecrypt(string publicKey, string text)
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, RSAOptions.Default);
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, string text)
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, RSAOptions.Default);
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.FromPrivateKey(bio))
100             {
101                 using (MessageDigestContext sha256 = new MessageDigestContext(Sha256))
102                 {
103                     byte[] msgByte = encoding.GetBytes(text);
104                     byte[] signByte = sha256.Sign(msgByte, cryptoKey);
105                     return Convert.ToBase64String(signByte);
106                 }
107             }
108         }
109     }
110
111     /// <summary>
112     /// 公钥验签
```

```
113     /// </summary>
114     public static bool Verify(string publicKey, string text,
115     {
116         using (BIO bio = new BIO(publicKey))
117         {
118             using (CryptoKey cryptoKey = CryptoKey.FromPubl:
119             {
120                 using (MessageDigestContext sha256 = new Mes
121                 {
122                     byte[] msgByte = encoding.GetBytes(text)
123                     byte[] signByte = Convert.FromBase64Stri
124                     return sha256.Verify(msgByte, signByte,
125                 }
126             }
127         }
128     }
129 }
```

标签: [OpenSSL.Net](#)

好文要顶

关注我

收藏该文



Azeri

关注 - 130

粉丝 - 4

+加关注

0

推荐

0

反对

« 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](#)

登录后才能发表评论, 立即 [登录](#) 或 [注册](#), [访问](#) 网站首页

【推荐】阿里云实时计算训练营重磅开启, 4天技能突破, 抢天猫精灵!

【推荐】阿里云春招即将开始, 提前下载面试宝典稳拿Offer

【推荐】阿里云Java训练营--就5天, 名师带你实战Spring Boot 2.5

【推荐】免费领最高6000元好云礼! 15种权益祝你运气爆棚

【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载!

【推荐】注册 Amazon Web Services(AWS) 账号, 成为博客园赞助者

【推荐】华为HMS Core Discovery直播间-七个推送技巧带你玩转App运营



AWS免费产品:

· [如何在AWS上免费构建网站](#)

- [AWS免费云存储解决方案](#)
- [在AWS上免费构建数据库](#)
- [AWS上的免费机器学习](#)



最新新聞：

- 國際知名AI學者陶大程出任京東探索研究院院長
  - 29歲網紅吃播“泡泡龍”去世曾靠“給自助餐廳上課”成名
  - 換電風波“劇終”！特斯拉工商變更：刪除“換電設施銷售”
  - 螞蟻森林價值首度公佈：5億人在手機上種樹種出113億
  - 影視劇中的“滴血認親”有科學依據嗎？終於明白了
- » 更多新聞...