大樹下玩耍

OpenSSL.Net使用隨記（三）

> 算法中DH與ECDH算法用來進行密鑰協商算法

- **DH**

```
1  class Program_DH
2      {
3          static  void Main2( string [] args)
4          {
5              GenKey(p, out aPrivateKey, out aPublicKey);
6
7             GenKey(p, out bPrivateKey, out bPublicKey);
8              bCompute = ComputeKey(p, bPrivateKey , bPublicKey, a
9
10             aCompute = ComputeKey(p, aPrivateKey, aPublicKey, bI
11
12            Console.WriteLine(aCompute == bCompute);
13              Console.ReadKey();
14          }
15
16         ///  <summary>
17         ///完全公開的P（質數）,G（底數）值
18         ///  </ summary>
19         static  string p = @" -----BEGIN DH PARAMETERS-----
20 MEYCQQCFO+ureuiANnvFOg79ojIyjVgdxuD4G7ERecH1xD+J7wDbgwZqejsTsg\
21 yE1aeTXiLvtGNcMLbwgGxkRT9S67AgEC
22 -----END DH PARAMETERS-----
23 " ;
24          // /  <summary>
25         /// A產生的3個常量
26         ///  </summary>
27         static  string aPrivateKey, aPublicKey, aCompute;
28
29         ///  <summary>
30         /// B產生的3個常量
31         ///  </summary>
32         static  string bPrivateKey, bPublicKey, bCompute;
33
34         ///  <summary>
35         ///生成DH算法的隨機數
36         ///  </summary>
37         ///  <param name="source"> P · G </param>
38         // /  <param name="privateKey">生成隱藏的隨機數</param>
39         /// <param name="publicKey">生成公開的隨機數</param>
40         public  static  void GenKey( string source, out  string
41          {
```

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

統計

隨筆- 5
文章- 0
評論- 0
閱讀- 6016

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

```
42              using (DH dhCrypto = DH.FromParameters(source)) / ,
43              {
44                  dhCrypto.GenerateKeys(); //生成隨機數
45                 privateKey = dhCrypto.PrivateKey.ToHexString();
46                  publicKey = dhCrypto.PublicKey.ToHexString();
47              }
48          }
49
50          ///  <summary>
51          ///計算DH算法的最終值（相同值）
52           ///  </summary>
53          ///  <param name="source"> P·G < /param>
54          ///  <param name="fromPrivateKey"> A方隱藏的隨機數</param
55          ///  <param name="fromPublicKey"> A方公開的隨機數</param>
56          // /  <param name="toPublicKey"> B方公開的隨機數</param>
57          ///  <returns>最終值（相同值）</returns>
58          public  static string ComputeKey( string source, string
59           {
60              using (DH dhCrypto = DH.FromParameters(source)) //
61              {
62                  dhCrypto.PrivateKey = BigNumber.FromHexString(
63                  dhCrypto.PublicKey = BigNumber.FromHexString(f
64                  byte [] keyBytes = dhCrypto.ComputeKey(BigNumbe
65                  returnBitConverter.ToString(keyBytes);
66              }
67          }
68      }
```

- **ECDH**

  生成兩組EC密鑰並相互公開公鑰即能完成ECDH算法

```
1  class Program_ECDH
2      {
3          static  void Main2( string [] args)
4          {
5              string aCompute = ComputeKey(aPrivateKey, bPublicK
6              string bCompute = ComputeKey(bPrivateKey, aPublicK
7              Console.WriteLine(aCompute == bCompute);
8              Console.ReadKey();
9          }
10
11          ///  <summary>
12          /// a方私鑰不公開
13          ///  </summary>
14         static  string aPrivateKey = @" -----BEGIN EC PARAMETERS-
15  BgUrgQQAIw==
16  -----END EC PARAMETERS-----
17  -----BEGIN EC PRIVATE KEY--- --
18  MIHcAgEBBEIAzb3CKEl2y87Q1dbqiOCG0UkBceI9V5nA4N0vXZx7xgJTHtfHCe
19  y/72GTZk7PQw89aTU7fdQ12NRC2hYiP2O1WgBwYFK4EEACOhgYkDgYYABAEwtG
20  5cGCineqYs3VPHdadOJgIwD0BGkuSEOWt3RD11S5OiBpY0bVYkYHKvySZYPfvF
21  EZOTNyNue3JZ0ubWzQDWHULl/P9t8LZrPrIMC43sHuoHDV0BhcsO/HUWKU9QBC
22  S++px6BwYrNoFaenJoHOVtDs8veqH1aAAQW1Mbb56A==
23  -----END EC PRIVATE KEY-----
```

```
" ;
        ///  <summary>
        /// a方公鑰對b方公開
        ///  </summary>
        static  string aPublicKey = @" -----BEGIN PUBLIC KEY----
MIGbMBAGByqGSM49AgEGBSuBBAAjA4GGAAQBMLRu0+XBgop3qmLN1Tx3WnTiYCN
9ARpLkhDlrd0Q9dUuTogaWNG1WJGByr8kmWD37xSFhGTkzcjbntyWdLm1s0A1h1
5fz/bfC2az6yDAuN7B7qBw1dAYXLDvx1FilPUAQmIUvvqcegcGKzaBWnpyaBzlk
7PL3qh9WgAEFtTG2+eg=
-----END PUBLIC KEY-----
" ;
         ///  <summary>
        /// b方私鑰不公開
        /// </summary>
        static  string bPrivateKey = @" -----BEGIN EC PARAMETERS
BgUrgQQAIw==
-----END EC PARAMETERS-----
-----BEGIN EC PRIVATE KEY-----
MIHcAgEBBEIBkm1mKzVrWaq0oSanR/45y7x6B+W8/PxymW2PCcc11azuZXusXC
48nXvM47Y02py1NsoDFK81EGUSokRVzKvC2gBwYFK4EEACOhgYkDgYYABADvLn
ai96mEX1PDcak0B4buXZjSlDgcMSNiPdAC7SaKwCHLvQXc+JCQkBQg8Bi6LNvz
q/DXKz5BpKYkIlmkuwDs6KlYlBoHTHI7hhneBcGAcrou5ay0+djFyaPcbCQgpS
z1Ot1nRz8nbqQW3PE7Cc/kB6eRQF4YWsjPiVBXbpiw==
-----END EC PRIVATE KEY-----
";
         ///  <summary>
        /// b方公鑰對a方公開
        ///  </summary>
        static  string bPublicKey = @" -----BEGIN PUBLIC KEY----
MIGbMBAGByqGSM49AgEGBSuBBAAjA4GGAAQA7y55MWovephF9Tw3GpNAeG7l2Y
Q4HDEjYj3QAu0misAhy70F3PiQkJAUIPAYuizb80uKvw1ys+QaSmJCJZpLsA7O
WJQaB0xyO4YZ3gXBgHK6LuWstPnYxcmj3GwkIKUsys9TrdZ0c/J26kFtzxOwnP
enkUBeGFrIz41QV26Ys=
-----END PUBLIC KEY-----
" ;

        ///  <summary>
        ///簽名回調
        ///  </summary>
        private  static  byte [] ComputeKeyHandler( byte [] mess
         {
             using (MessageDigestContext hashDigest = new Messag
             {
                 return hashDigest.Digest(message);
             }
         }

        ///  <summary>
        ///簽名算法
        ///  </summary>
        private  staticMessageDigest HashDigest
         {
             get { return MessageDigest.SHA256; }
         }

        ///  <summary>
        ///計算最終值
        ///  </summary>
```

```
82          ///  <param name="fromPrivateKey"> a方私鑰</param>
83          ///  <param name="toPublicKey"> b方公鑰</param>
84          ///  <returns>最終值</returns>
85          public  static  string ComputeKey( string fromPrivateKey
86           {
87               using (CryptoKey toCryptoKey = CryptoKey.FromPublic
88               {
89                   using (Key toKey = toCryptoKey.GetEC())
90                   {
91                       using (CryptoKey fromCryptoKey = CryptoKey.
92                       {
93                           using (Key fromKey = fromCryptoKey.GetK
94                           {
95                               byte [] buffer = new  byte[HashDige
96                               int aout = fromKey.ComputeKey(toKey
97                               return BitConverter.ToString(buffer
98                           }
99                       }
100                  }
101              }
102          }
103
104      }
```

標籤: OpenSSL.Net

Azeri
關注- 130
粉絲- 4
+加關注

«上一篇： OpenSSL.Net使用隨記（二）
»下一篇： OpenSSL.Net使用隨記（四）

posted on 2018-05-03 20:21 Azeri 閱讀(304)評論(0) 編輯 收藏

刷新評論 刷新頁面 返回頂部