[**关于PKCS5Padding与PKCS7Padding的区别**](http://www.cnblogs.com/AloneSword/p/3491466.html)

工作中，我们常常会遇到跨语言平台的加密解密算法的交互使用，特别是一些标准的加解密算法，都设计到数据块Block与填充算法的  
问题，例如C#与JAVA中的常见的填充算法如下：

**.Net中的填充算法：**

|  |  |
| --- | --- |
| **成员名称** | **说明** |
| **ANSIX923** | ANSIX923 填充字符串由一个字节序列组成，此字节序列的最后一个字节填充字节序列的长度，其余字节均填充数字零。  下面的示例演示此模式的工作原理。假定块长度为   8，数据长度为 9，则填充用八位字节数等于 7，数据等于 FF FF FF FF FF FF FF FF FF：  数据： FF FF FF FF FF FF FF FF FF  X923 填充： FF FF FF FF FF FF FF FF FF 00 00 00 00 00 00 07 |
| **ISO10126** | ISO10126 填充字符串由一个字节序列组成，此字节序列的最后一个字节填充字节序列的长度，其余字节填充随机数据。  下面的示例演示此模式的工作原理。假定块长度为 8，数据长度为 9，则填充用八位字节数等于 7，数据等于 FF FF FF FF FF FF FF FF FF：  数据： FF FF FF FF FF FF FF FF FF  ISO10126 填充： FF FF FF FF FF FF FF FF FF 7D 2A 75 EF F8 EF 07 |
| **None** | 不填充。 |
| **PKCS7** | PKCS #7 填充字符串由一个字节序列组成，每个字节填充该字节序列的长度。  下面的示例演示这些模式的工作原理。假定块长度为 8，数据长度为 9，则填充用八位字节数等于 7，数据等于 FF FF FF FF FF FF FF FF FF：  数据： FF FF FF FF FF FF FF FF FF  PKCS7 填充： FF FF FF FF FF FF FF FF FF 07 07 07 07 07 07 07 |
| **Zeros** | 填充字符串由设置为零的字节组成。 |

**JAVA中支持的填充算法（Cipher）有**

| **Alg. Name** | **Description** |
| --- | --- |
| NoPadding | No padding. |
| ISO10126Padding | This padding for block ciphers is described in [5.2 Block Encryption Algorithms](http://www.w3.org/TR/xmlenc-core/#sec-Alg-Block) in the W3C's "XML Encryption Syntax and Processing" document. |
| OAEPPadding, OAEPWith<digest>And<mgf>Padding | Optimal Asymmetric Encryption Padding scheme defined in PKCS1, where <digest> should be replaced by the message digest and <mgf> by the mask generation function. Examples: **OAEPWithMD5AndMGF1Padding** and **OAEPWithSHA-512AndMGF1Padding**.   If **OAEPPadding** is used, **Cipher** objects are initialized with a **javax.crypto.spec.OAEPParameterSpec** object to suppply values needed for OAEPPadding. |
| PKCS1Padding | The padding scheme described in [PKCS1](http://www.rsasecurity.com/rsalabs/pkcs), used with the RSA algorithm. |
| PKCS5Padding | The padding scheme described in [RSA Laboratories, "PKCS5: Password-Based Encryption Standard," version 1.5, November 1993](http://www.rsasecurity.com/rsalabs/pkcs). |
| SSL3Padding | The padding scheme defined in the SSL Protocol Version 3.0, November 18, 1996, section 5.2.3.2 (CBC block cipher):  block-ciphered struct {  opaque content[SSLCompressed.length];  opaque MAC[CipherSpec.hash\_size];  uint8 padding[  GenericBlockCipher.padding\_length];  uint8 padding\_length;  } GenericBlockCipher;  The size of an instance of a GenericBlockCipher must be a multiple of the block cipher's block length.   The padding length, which is always present, contributes to the padding, which implies that if:  sizeof(content) + sizeof(MAC) % block\_length = 0,  padding has to be (block\_length - 1) bytes long, because of the existence of **padding\_length**.   This make the padding scheme similar (but not quite) to PKCS5Padding, where the padding length is encoded in the padding (and ranges from 1 to block\_length). With the SSL scheme, the sizeof(padding) is encoded in the always present **padding\_length** and therefore ranges from 0 to block\_length-1. |

简单对比之下发现，通用的有**None**，**ISO10126**两种填充法，实际上PKCS5Padding与PKCS7Padding基本上也是可以通用的。

通过研读参考资料下面的参考资料可以发现两者定义的区别：

* + [*Def*] *PKCS #7: Cryptographic Message Syntax Standard*,  
    An RSA Laboratories Technical Note, Version 1.5. Revised November 1, 1993. [http://www.cnblogs.com/midea0978/admin/ftp://ftp.rsa.com/pub/pkcs/ascii/pkcs-7.asc](http://www.cnblogs.com/midea0978/admin/ftp:/ftp.rsa.com/pub/pkcs/ascii/pkcs-7.asc)
  + [*Inf*] *PKCS #5: Password-Based Encryption Standard*,  
    An RSA Laboratories Technical Note, Version 1.5. Revised November 1, 1993. [http://www.cnblogs.com/midea0978/admin/ftp://ftp.rsa.com/pub/pkcs/ascii/pkcs-5.asc](http://www.cnblogs.com/midea0978/admin/ftp:/ftp.rsa.com/pub/pkcs/ascii/pkcs-5.asc)

在PKCS5Padding中，明确定义Block的大小是8位，而在PKCS7Padding定义中，对于块的大小是不确定的，可以在1-255之间（块长度超出255的尚待研究），填充值的算法都是一样的：

value=k - (l mod k)  ,K=块大小，l=数据长度，如果l=8, 则需要填充额外的8个byte的8

在.net中，例如TripleDESCryptoServiceProvider ，默认BlockSize=64bits=8bytes，所以在这种情况下在PKCS5Padding=PKCS7Padding。

如果在C#中自己定义了一个不是64bits的加密块大小，同时使用PKCS7Padding，那么在java中使用JDK标准的PKCS5Padding就不能解密了。

JAVA Code示例

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | try {      byte[] KEY\_DATA = {1,8,-49,-31,77,90,10,121,-14,109,107,38,29,68,59,5,82,49,31,42,-25,67,96,15};      Cipher cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");      SecretKeySpec key = new SecretKeySpec(KEY\_DATA, "DESede");//生成加密解密需要的Key      cipher.init(Cipher.ENCRYPT\_MODE, key);      byte[] res = cipher.doFinal(data.getBytes());  } catch (Exception e) {      e.printStackTrace();  } |



C# Code示例\

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | TripleDESCryptoServiceProvider des = new TripleDESCryptoServiceProvider();  des.Mode=CipherMode.ECB;  des.Padding=PaddingMode.PKCS7;  byte[] buffer =Encoding.Default.GetBytes("明文");  MemoryStream stream = new MemoryStream();  byte[] key=Convert.FromBase64String("AQjP4U1aCnnybWsmHUQ7BVIxHyrnQ2AP");  CryptoStream encStream = new CryptoStream(stream, des.CreateEncryptor(key, null), CryptoStreamMode.Write);  encStream.Write(buffer, 0, buffer.Length);  encStream.FlushFinalBlock();  byte[] res=stream.ToArray();  Console.WriteLine("result:"+Convert.ToBase64String(res)); |

* + 参考<http://www.users.zetnet.co.uk/hopwood/crypto/scan/cs.html#pad_PKCSPadding>

<http://www.cnblogs.com/midea0978/articles/1437257.html>