Base64 之逆编码表

1. 简介

Base64 的逻辑是：字符串>>每个字符的 8 位二进制连接>>每 6 位转换为十进制对应编码表连接转换后字符；如果要编码的字节数不能被 3 整除，最后会多出 1 个或 2 个字节，那么可以使用下面的方法进行处理：先使用 0 字节值在末尾补足，使其能够被 3 整除，然后再进行 base64 的编码。在编码后的 base64 文本后加上一个或两个'='号，代表补足的字节数。

由于 6 位二进制最大为 111111 即 63 最小为 000000 所以编码表中共有 64 个字符。

Base 标准编码表：

2. 示例

正常情况下 Base64 的编码表是：

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=，但如果将编码表中的字符换其它顺序排一下结果又是什么样子呢？

我使用了这个编码表(等同于 key)：

abcdefghijklmnopqrstuvwxyz0123456789+/=ABCDEFGHIJKLMNOPQRSTUVWXYZ，编码字符：I Love You!

得到字符串 :ssbm1Qz/if/I3se= ,但使用标准编码表解码得到 :ЯՈЙ| 几个"乱码"…… 那么在已知编码前字符和编码后字符但编码表不知的情况下能不能根据数组对照来确定编码表呢？

答案也是肯定的：首先根据明文和密文的长度来确定使用的有没有可能是Base64 编码(详情 Baidu、Google)，然后再将明文转换为 8 位二进制每 6 位再转为十进制。

假如我们有明文：11CA467C5B1C3C0AB1D6C8A81104CC868CDC0A91 和密文：mtfdqtq2n0m1qJfdm0mWquiXrdzdoee4mteWnendody4q0rdmee5mq==

那确定编码表的过程如下：

Step1:明文转为 8 位二进制：

00110001001100010100001101000001001101000011011000110111010000110011010101000010001100010100001100110011010000110011000001000000101000010001100010100010000110110010000110011110000100000100111000001100010011000100110000001101000100001101000011001110000011011000111000010000110100010001000011001100000100000100111001001100001

Step2:每 6 位二进制数转为十进制(我以|分割便于查看)：

12|19|5|3|16|19|16|54|13|52|12|53|16|35|5|3|12|52|12|48|16|20|8|49|17|3|25|3|14|4|4|56|12|19|4|48|13|4|13|3|14|3|24|56|16|52|17|3|12|4|4|57|12|16|

Step3 对应加密后字串：

mtfdqtq2n0m1qJfdm0mWquiXrdzdoee4mteWnendody4q0rdmee5mq==

得到 m 在编码表中的位置是 12，t 在编码表中的位置是 19，f 在编码表中的位置是 5……(因后面的 m 都是 12，t 都是 19……，所以可以确定这是换了编码表的 Base64)。在使用了多组明文和密文对照之后得到了变异的编码表：

abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/=

至此成功完成了编码表的逆算。

对于一次 Base64 编码的的运算可以通过几组对照的明、密文轻松逆得编码表，假如是两次使用同编码表的 Base64 编码，还有没有可能得到编码表呢？

可以做如下分析 :(简单的举例 ,因第一个字符编码后不受后面字符的影响，所以下面分析的是每一次编码后的第一个字符)

第一次编码后字符空间位于编码表的 8 位至 31 位(可打印字符的 ASSCII 码为 32-126)

```
……
Bin(A)= 01000001  Base64(A)=编码表中第 16 位
Bin(E)= 01000101  Base64(E)=编码表中第 17 位
Bin(I)= 01001001  Base64(I)=编码表中第 18 位
Bin(M)= 01001101  Base64(M)=编码表中第 19 位
Bin(Q)= 01010001  Base64(Q)=编码表中第 20 位
Bin(U)= 01010101  Base64(U)=编码表中第 21 位
Bin(Y)= 01011001  Base64(Y)=编码表中第 22 位
……
```

由于第一次编码后字符范围一定在下面这些字符范围中：

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/=
```

其中 ASSCII 码最小的为”43” 最大的为”122”

所以二次编码后字符空间位于编码表的第 10 位至 30 位。

由于二次编码的结果是知道的，所以可以用多组密文来确定编码表位于 10-30 之间的所有字符(但顺序是不知道的)。

```
……
```

Base64(Base64(A))= Base64(编码表中第 16 位的字符)

Base64(Base64(E))= Base64(编码表中第 17 位的字符)

Base64(Base64(I))= Base64(编码表中第 18 位的字符)

……

这样编码表 10-30 位所有字符的 Base 编码结果都知道了(结果字符还位于 10 至 30 位之间)

由此得到了一个 21 元的方程(如果解的出来那么编码表中的一部分字符的位置就确定了，不过还没有尝试能不能解出来……)

可见如果经过了二次编码，要还原编码表还是很困难的