**加密格式和算法**

# 基本算法

核心算法采用异或算法，所以加密密钥和解密密钥是一样的。

加密过程和解密过程也是一样的，对字符进行XOR异或运算就可以了。

1 ^ 5 = 4

4 ^ 5 = 1

# 密钥

密钥分为代码密钥和文件密钥。

代码密钥是一个长度为4-16的ascii字符串，代码密钥保存在代码中。

文件密钥是一个长度为16-32的二进制数组，文件密钥在每次保存文件的时候会计算出来，参考文件密钥生成算法文档。

# 包格式

加密后的包分为包头和包体。

包头为一些相关信息，包体为加密后的内容格式。

# 包头

包头格式如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 识别码 | 版本号 | 文件长度 | 校验码 | 文件密钥长度 | 文件密钥内容 | 包体长度 | 加密长度 |

## 识别码

识别码固定为4个字节，内容为“BGT1”

## 版本号

版本号为4字节整数，采用小端模式，数据的低位保存在低地址中，数据的高位保存在高地址中。

版本号由1开始增加，每一个版本号对应一个唯一的代码密钥。

代码中需要保存所有的版本号和密钥的对应关系。

当代码发现文件中版本号比自己的版本号要高时，直接解析失败。

## 文件长度

文件长度为4字节整数，采用小端模式，数据的低位保存在低地址中，数据的高位保存在高地址中。

## 校验码

校验码为1个字节，采用对整个文件的字节进行异或（XOR）运算得到的结果。

注意，校验码是计算除了本字节之外的文件内容。

## 文件密钥长度

文件密钥长度为加密字段。

文件密钥长度为1个字节整数，采用真实长度与代码密钥每一位进行XOR运算。

假设代码密钥为“ABCDEFGH”，文件密钥真实长度为20；

则加密后的文件密钥长度=20 ^ ’A’ ^ ‘B’ ^ ‘C’ ^ ‘D’ ^ ’E’ ^ ‘F’ ^ ‘G’ ^ ‘H’

## 文件密钥内容

文件密钥内容为加密字段。

文件密钥的长度在前一个算法中获得，加密方式为XOR运算。

假设文件密钥为{1,3,5,7,9,2,4,6,8,0,11,13,15,17,19,12}，长度为16；

代码密钥为“ABCDEFGH”，长度为8。

则加密后的文件密钥如下：

第一个字节=1 ^ 16 ^ ‘A’;

第二个字节=3 ^ 16 ^ ‘B’;

……

第八个字节=6 ^ 16 ^ ‘H’;

第九个字节=8 ^ 16 ^ ‘A’; //回到代码密钥头部

第十个字节=0 ^ 16 ^ ‘B’;

……

第十六个字节=12 ^ 16 ^ ‘H’;

## 包体长度

包体长度为加密字段。

包体长度为4个字节整数，采用小端模式，数据的低位保存在低地址中，数据的高位保存在高地址中。

假设文件密钥为{1,3,5,7,9,2,4,6,8,0,11,13,15,17,19,12}，长度为16；

代码密钥为“ABCDEFGH”，长度为8。

包体长度为4567。

则加密后的包体长度如下：

第一个字节= (4567 & 0xff) ^ 1 ^ ‘A’;

第二个字节= (4567 >> 8 & 0xff) ^ 3 ^ ‘B’;

第三个字节= (4567 >> 16 & 0xff) ^ 5 ^ ‘C’;

第四个字节= (4567 >> 24 & 0xff) ^ 7 ^ ‘D’;

包体长度用来校验文件长度和加密方式是否正常。

## 加密长度

加密长度为加密字段。

加密长度为4个字节整数，采用小端模式，数据的低位保存在低地址中，数据的高位保存在高地址中。

假设文件密钥为{1,3,5,7,9,2,4,6,8,0,11,13,15,17,19,12}，长度为16；

代码密钥为“ABCDEFGH”，长度为8。

加密长度为2048。

则加密后的包体长度如下：

第一个字节= (2048 & 0xff) ^ 1 ^ ‘A’;

第二个字节= (2048 >> 8 & 0xff) ^ 3 ^ ‘B’;

第三个字节= (2048 >> 16 & 0xff) ^ 5 ^ ‘C’;

第四个字节= (2048 >> 24 & 0xff) ^ 7 ^ ‘D’;

加密长度用来加密指定长度的内容，剩余内容不进行加密，直接复制。

加密长度必须大于或等于0，如果加密长度>包体长度，则计算时需要按min（包体长度，加密长度）来进行计算。

# 包体

包体为加密后的内容，与加密前的内容相比，长度是一样的。

加密过程涉及到以下几点。

## 密钥变换

在加密过程中，文件密钥和代码密钥都需要变换。

比如原始内容为“abcdef1234567890ABCDEF”；

密钥为{1, 3, 5, 7}，长度为4；

则第1个字节= ‘a’ ^ 1;

2 = ‘b’ ^ 3;

3 = ‘c’ ^ 5;

4 = ‘d’ ^ 7;

第五个字节开始，由于密钥已经用过一次，所以需要进行密钥变换。

密钥使用次数 = 1;

密钥变换下标 = 1 % 4 = 1;

密钥变化量 = (1 \* 7) & 0xff = 7; //这里的1是密钥使用次数

密钥值 = (3 + 7) & 0xff = 10; //这里的密钥值3和10可能是负数

所以密钥改为{1, 10, 5, 7}，继续进行计算。

第5个字节= ‘e’ ^ 1;

6 = ‘f’ ^ 10;

7 = ‘1’ ^ 5;

8 = ‘2’ ^ 7;

第八个字节开始，由于密钥已经用过二次，所以需要进行密钥变换。

密钥使用次数 = 2;

密钥变换下标 = 2 % 4 = 2;

密钥变化量 = (2 \* 7) & 0xff = 14; //这里的2是密钥使用次数

密钥值 = (5 + 14) & 0xff = 19; //这里的密钥值5和19可能是负数

所以密钥改为{1, 10, 19, 7}，继续进行计算。

……

## 内容加密

加密内容的过程中，需要把文件密钥和代码密钥同时加入运算。

比如原始内容为“abcdef1234567890ABCDEF”；

假设文件密钥为{1,3,5,7,9,2,4,6,8,0,11,13,15,17,19,12}，长度为16；

代码密钥为“ABCDEFGH”，长度为8。

那么第1个字节 = a ^ 1 ^ ‘A’;

2 = b ^ 3 ^ ‘B’

……

9 = 3 ^ 8 ^ ‘A’

……

## 整体流程

整体加密流程就是结合密钥变换和内容加密，得到最终的结果。