读书成诗解密题分析 一

作者：Jireh
原文链接：https://ld246.com/article/1604026055105
来源网站：链滴

读书成诗解密题分析 一

读书成诗这个公众号估计很多人应该都知道吧，出过JetBrains 全家桶破解的工具，但是获取激活工具要先解谜题

虽然我已经激活了（正版），但是解密话的还是蛮有意思的

我的激活途径（免费），有两个。一个是学生的账号可以免费使用，还有一个是在Github上开源了一项目，去申请后JetBrains客服会赠送全家桶的授权Key，这两个途径每年都需要重新获取授权。以上种途径，自行百度搜索，会有很多资料的，以后有机会话，我再写相应的教程。

避免公开讨论导致途径关闭，不会直接将最终结果给出，也望各位低调，不要将最终结果公开

虽然我不需要（手动滑稽smirk），但是其他人
能会需要

希望大家支持正版 支持正版 支持正版（重要的事情说3遍）

解密分析

废话有点多接下来，进入正题。到2020年10月28日 17:24:01为止最新的谜题为

黑白皆算，对我等众猿而言中央C所在位置数优剃爱肤杠吧爱慕帝贰亿次的值是?

1. 首先需要知道算什么【黑白皆算，对我等众猿而言中央C】。

中央C，嗯 一看就知道是钢琴的键位了。要求**中央C的位置**，**黑白皆算**，就是黑白键位都算进去。估很多人去百度上搜中央C的位置，不过要注意了，百度上会给你49这个结果，但是这个**49是中央C在系体系中的位置**。其实可以去晚上找钢琴的图，或者家里有钢琴的可以数一下位置（我家里有钢琴mirk），答案是 **40**，但是题目中有说**对我等众猿而言**，计算机中**数据都是0开始**的，所以**最终案是39**

2. 最后就是算法 **优剃爱肤杠吧爱慕帝贰亿次。**

优剃爱肤杠吧=UTF-8

爱慕帝=MD2

所以是把39编码转换成utf-8在md2加密一亿次就是最终结果了

最后上代码

解密代码

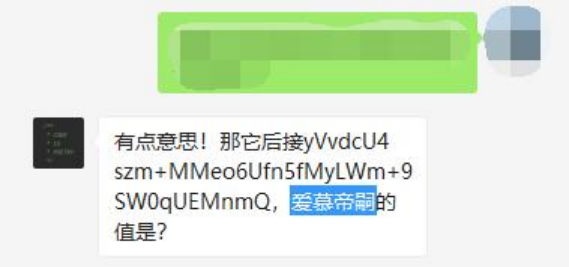
Java

```
String data = "39";
try {
    for (int i = 1; i <= 100000000; i++) {
        data = new String(DigestUtils.md2Hex(data).getBytes("gbk"), "utf-8");
    }
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
```

```
}
System.out.println(data);
```

Python

```
# -*- coding:utf-8 -*-
from Crypto.Hash import MD2
# 原文
data = "39"
for i in range(100000000):
    data = MD2.new(data.encode("utf8")).hexdigest()
print(data)
```



最后拿着解密结果，发送到公众号试试，回复下一步的加密步骤了

接下就是把之前的结果，拼接yVvdcU4szm+MMeo6Ufn5fMyLWm+9SW0qUEMnmQ后在进行MD加密，就可以了

以下在附上MD4加密算发，如果觉得麻烦网上也有在线MD4加密的网站https://www.qqxiuzi.cn/bianma/md4.htm

MD4算法

```
public class MD4 {

    private static int A, B, C, D;

    private static int X[] = new int[16];

    private static int F(int X, int Y, int Z) {
        return (X & Y) | ((~X) & Z);
    }

    private static int G(int X, int Y, int Z) {
        return (X & Y) | (X & Z) | (Y & Z);
    }

    private static int H(int X, int Y, int Z) {
        return X ^ Y ^ Z;
    }

    private static int lshift(int x, int s) {
        if (s == 0) {
```

```
        return x;
    }
    return (((x << s) & 0xFFFFFFFF) | ((x >> (32 - s)) & (0x7FFFFFFF >> (31 - s))));
}

private static int ROUND1(int a, int b, int c, int d, int k, int s) {
    return (lshift(a + F(b, c, d) + X[k], s));
}

private static int ROUND2(int a, int b, int c, int d, int k, int s) {
    return (lshift(a + G(b, c, d) + X[k] + (int) 0x5A827999, s));
}

private static int ROUND3(int a, int b, int c, int d, int k, int s) {
    return (lshift(a + H(b, c, d) + X[k] + (int) 0x6ED9EBA1, s));
}

public static void mdfour64(int M[]) {
    int j;
    int AA, BB, CC, DD;

    for (j = 0; j < 16; j++) {
        X[j] = M[j];
    }

    AA = A;
    BB = B;
    CC = C;
    DD = D;

    A = ROUND1(A, B, C, D, 0, 3);
    D = ROUND1(D, A, B, C, 1, 7);
    C = ROUND1(C, D, A, B, 2, 11);
    B = ROUND1(B, C, D, A, 3, 19);
    A = ROUND1(A, B, C, D, 4, 3);
    D = ROUND1(D, A, B, C, 5, 7);
    C = ROUND1(C, D, A, B, 6, 11);
    B = ROUND1(B, C, D, A, 7, 19);
    A = ROUND1(A, B, C, D, 8, 3);
    D = ROUND1(D, A, B, C, 9, 7);
    C = ROUND1(C, D, A, B, 10, 11);
    B = ROUND1(B, C, D, A, 11, 19);
    A = ROUND1(A, B, C, D, 12, 3);
    D = ROUND1(D, A, B, C, 13, 7);
    C = ROUND1(C, D, A, B, 14, 11);
    B = ROUND1(B, C, D, A, 15, 19);

    A = ROUND2(A, B, C, D, 0, 3);
    D = ROUND2(D, A, B, C, 4, 5);
    C = ROUND2(C, D, A, B, 8, 9);
    B = ROUND2(B, C, D, A, 12, 13);
    A = ROUND2(A, B, C, D, 1, 3);
    D = ROUND2(D, A, B, C, 5, 5);
    C = ROUND2(C, D, A, B, 9, 9);
```

```
    B = ROUND2(B, C, D, A, 13, 13);
    A = ROUND2(A, B, C, D, 2, 3);
    D = ROUND2(D, A, B, C, 6, 5);
    C = ROUND2(C, D, A, B, 10, 9);
    B = ROUND2(B, C, D, A, 14, 13);
    A = ROUND2(A, B, C, D, 3, 3);
    D = ROUND2(D, A, B, C, 7, 5);
    C = ROUND2(C, D, A, B, 11, 9);
    B = ROUND2(B, C, D, A, 15, 13);

    A = ROUND3(A, B, C, D, 0, 3);
    D = ROUND3(D, A, B, C, 8, 9);
    C = ROUND3(C, D, A, B, 4, 11);
    B = ROUND3(B, C, D, A, 12, 15);
    A = ROUND3(A, B, C, D, 2, 3);
    D = ROUND3(D, A, B, C, 10, 9);
    C = ROUND3(C, D, A, B, 6, 11);
    B = ROUND3(B, C, D, A, 14, 15);
    A = ROUND3(A, B, C, D, 1, 3);
    D = ROUND3(D, A, B, C, 9, 9);
    C = ROUND3(C, D, A, B, 5, 11);
    B = ROUND3(B, C, D, A, 13, 15);
    A = ROUND3(A, B, C, D, 3, 3);
    D = ROUND3(D, A, B, C, 11, 9);
    C = ROUND3(C, D, A, B, 7, 11);
    B = ROUND3(B, C, D, A, 15, 15);

    A += AA;
    B += BB;
    C += CC;
    D += DD;

    A &= 0xFFFFFFFF;
    B &= 0xFFFFFFFF;
    C &= 0xFFFFFFFF;
    D &= 0xFFFFFFFF;
}

public static void copy64(int M[], byte in[], int offset) {
    int i;
    for (i = 0; i < 16; i++) {
        M[i] = ((in[offset + i * 4 + 3] << 24) & 0xFF000000)
               | ((in[offset + i * 4 + 2] << 16) & 0xFF0000)
               | ((in[offset + i * 4 + 1] << 8) & 0xFF00)
               | (((int) in[offset + i * 4 + 0]) & 0xFF);
    }
}

public static void copy64(int M[], byte in[]) {
    copy64(M, in, 0);
}

public static void copy4(byte out[], int offset, int x) {
    out[offset] = (byte) (x & 0xFF);
```

```
    out[1 + offset] = (byte) ((x >> 8) & 0xFF);
    out[2 + offset] = (byte) ((x >> 16) & 0xFF);
    out[3 + offset] = (byte) ((x >> 24) & 0xFF);
}

public static byte[] mdfour(byte in[]) {
    byte out[] = new byte[16];
    byte buf[] = new byte[128];
    int n = in.length;
    int M[] = new int[16];
    int b = n * 8;
    int i;
    int offset;

    A = 0x67452301;
    B = 0xefcdab89;
    C = 0x98badcfe;
    D = 0x10325476;

    offset = 0;
    while (n > 64) {
        copy64(M, in, offset);
        mdfour64(M);
        offset += 64;
        n -= 64;
    }

    for (i = 0; i < 128; i++) {
        buf[i] = (i + offset < in.length) ? in[offset + i] : 0;
    }
    buf[n] = (byte) 0x80;

    if (n <= 55) {
        copy4(buf, 56, b);
        copy64(M, buf);
        mdfour64(M);
    } else {
        copy4(buf, 120, b);
        copy64(M, buf);
        mdfour64(M);
        copy64(M, buf, 64);
        mdfour64(M);
    }

    for (i = 0; i < 128; i++) {
        buf[i] = 0;
    }
    copy64(M, buf);

    copy4(out, 0, A);
    copy4(out, 4, B);
    copy4(out, 8, C);
    copy4(out, 12, D);
```

```
        A = B = C = D = 0;
        return out;
    }

    private static final char[] HEX_DIGITS = { '0', '1', '2', '3', '4', '5',
            '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f' };

    public static String toHexString(byte[] b) {
        return toHexString(b, 0, b.length);
    }

    public static String toHexString(byte[] b, int off, int len) {
        char[] buf = new char[len * 2];
        for (int i = 0, j = 0, k; i < len;) {
            k = b[off + i++];
            buf[j++] = HEX_DIGITS[(k >>> 4) & 0x0F];
            buf[j++] = HEX_DIGITS[k & 0x0F];
        }
        return new String(buf);
    }

    public static String MD4(String s) {
        return toHexString(mdfour(s.getBytes()));
    }
}
```

调用方法为

```
String data = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"+"yVvdcU4szm+MMeo6Ufn5fMyLWm+9SW
qUEMnmQ";
data= MD4.MD4(data);
System.out.println(data);
```

最后附上最终成功解密结果

读书成诗

牛！缘分到了，感谢相伴：

** 可回复 安装参数 获取安装参数。**
请一定低调使用，切勿外发!!!

原文链接：读书成诗解密题分析 一

是：README 是必须要看的