フラグの恋 出题人WriteUp by 圆周率

选手视角

旮旯给木基本素养部分

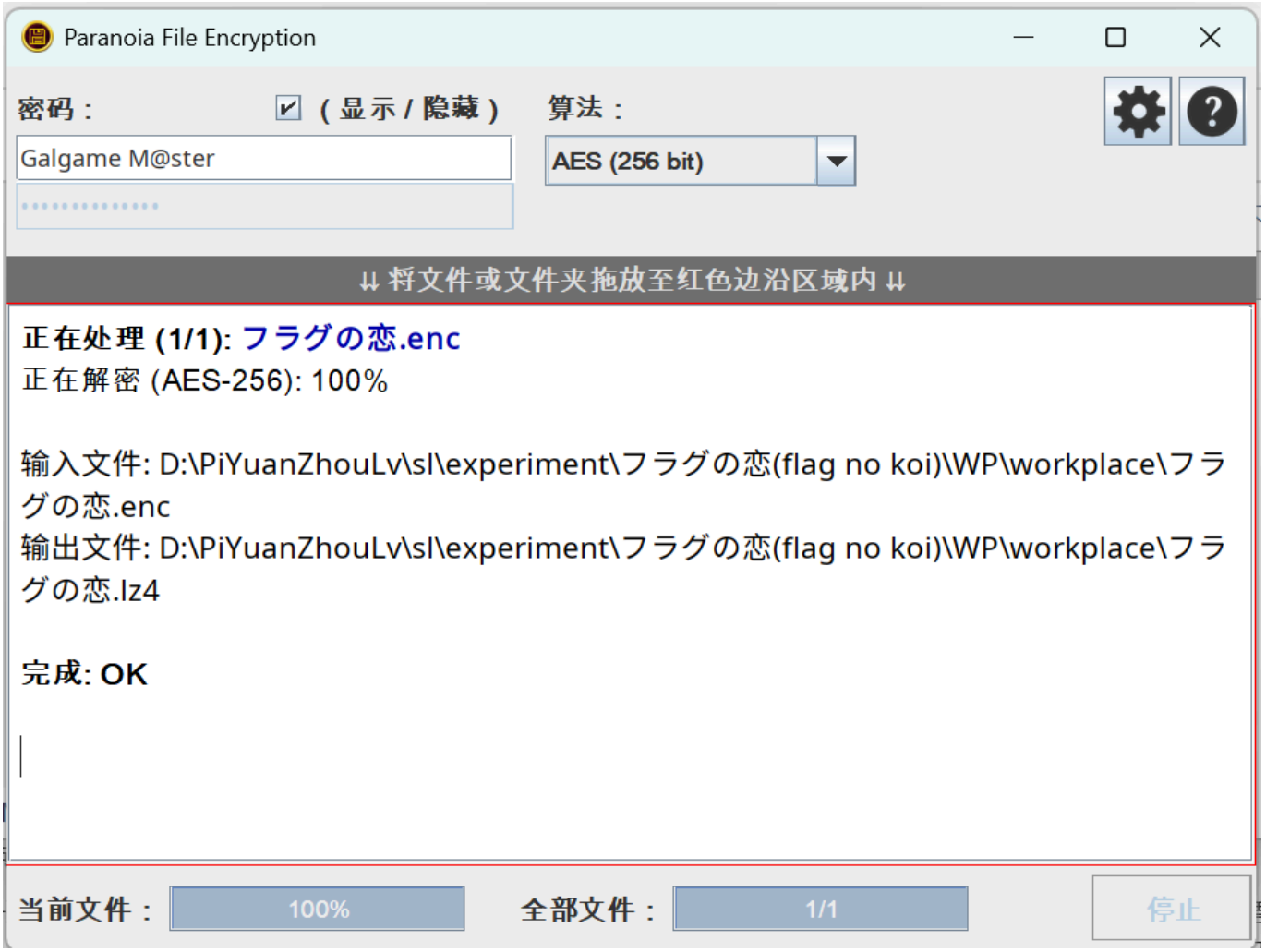
首先题目并没有附件和环境，而是给了一串神秘文字，分析文字，发现仅由四个字组成，由此想到是兽音加密，搜索“兽音译者”即可找到解密程序

兽音译者需要一个四个字符的字符表，只要稍微进行过网上冲浪就马上能知道大概率是“鸡你太美”，由此可以解出附件的网盘地址



接下来就是下载附件，得到一个.enc文件，经过简单检索，就可以找到一大堆教你解密的视频（看得出来，为了让大家都玩得了旮旯给木，还是有不少 ~~被新手气坏子的人~~ 好心人的）

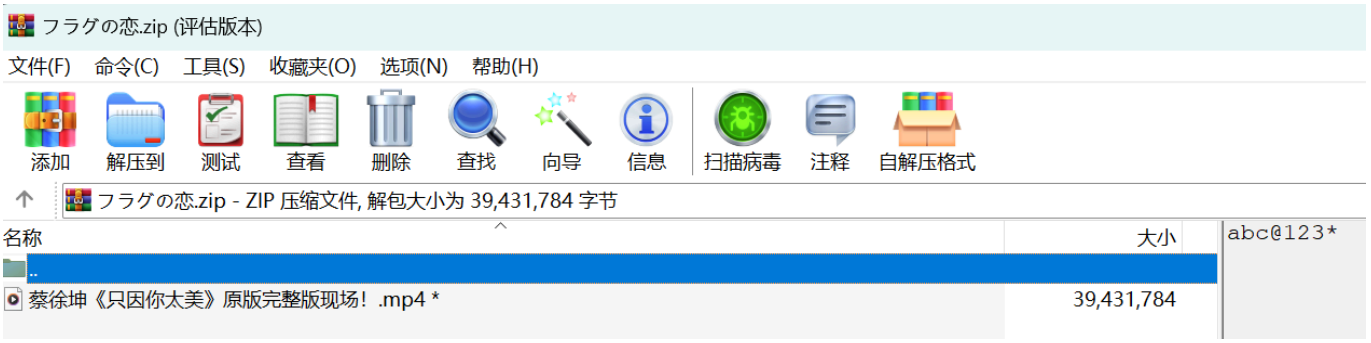
下载一个PTE，然后就可以使用密码解压了



接下来是一个.lz4文件，LZ4是一种压缩算法，可以使用Python解压

```
import lz4.frame
open('フラグの恋', 'wb').write(lz4.frame.decompress(open('フラグの恋.lz4', 'rb').read()))
```

接下来可以根据文件头判断，这是一个zip文件，修改扩展名后打开

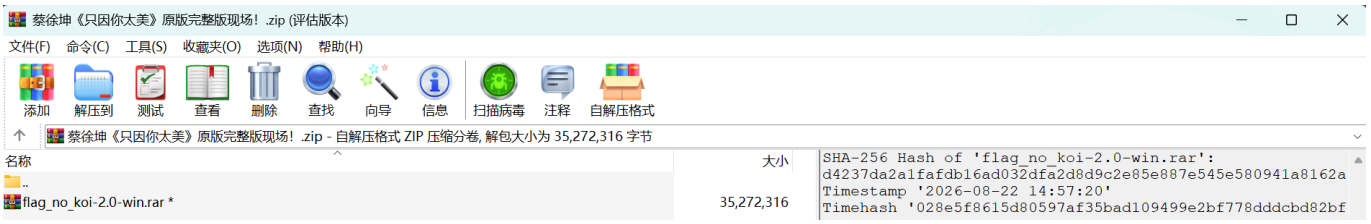


可以看见右侧注释区直接给出了密码，得到下一层文件

拿到后binwalk一下，发现其中含有一个zip文件，改扩展名后发现需要密码

```
└─$ binwalk 蔡徐坤《只因你太美》原版完整版现场！.mp4

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
1069829       0x105305        JBOOT STAG header, image id: 15, timestamp 0xCD0EE913, image size: 2342341097 bytes, image
 JBOOT checksum: 0x9EF7, header JBOOT checksum: 0x115B
4140471       0x3F2DB7        YAFFS filesystem root entry, big endian, type symlink, v1 root directory
4140607       0x3F2E3F        Zip archive data, encrypted at least v2.0 to extract, compressed size: 35272344, uncompres
sed size: 35272316, name: flag_no_koi-2.0-win.rar
39413332      0x2596654       xz compressed data
39421530      0x259865A       xz compressed data
```

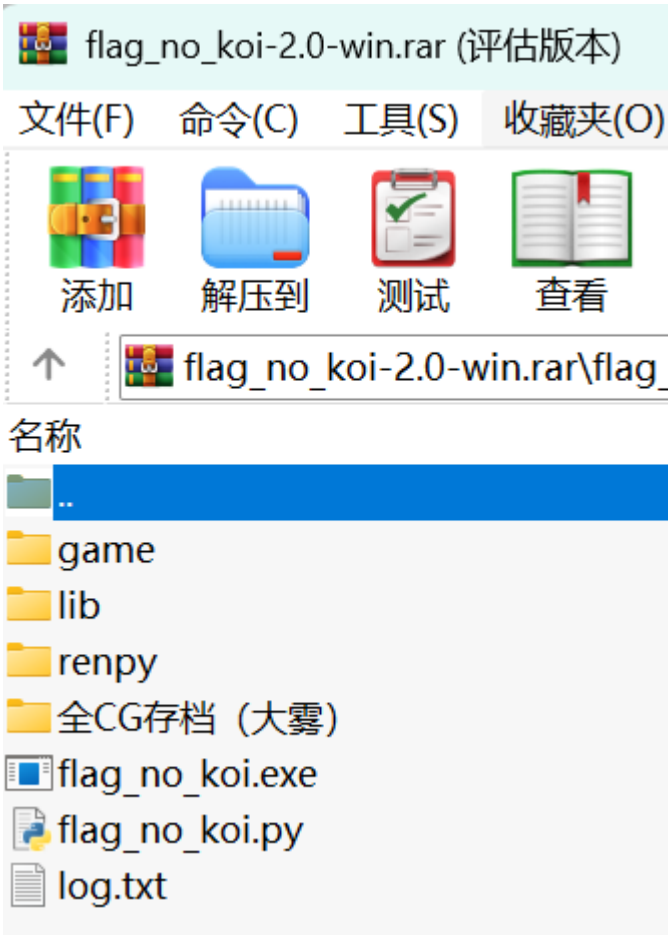


把扩展名改回来，考虑密码藏在视频里，视频不外乎就是图像和音频（当然，部分多媒体格式也支持嵌入文件，可以ffprobe一下），这里我们盯帧，发现22.44s附近有一个二维码



识别后得到“pwd is 114514”，说明压缩包密码是114514，得到下一层文件

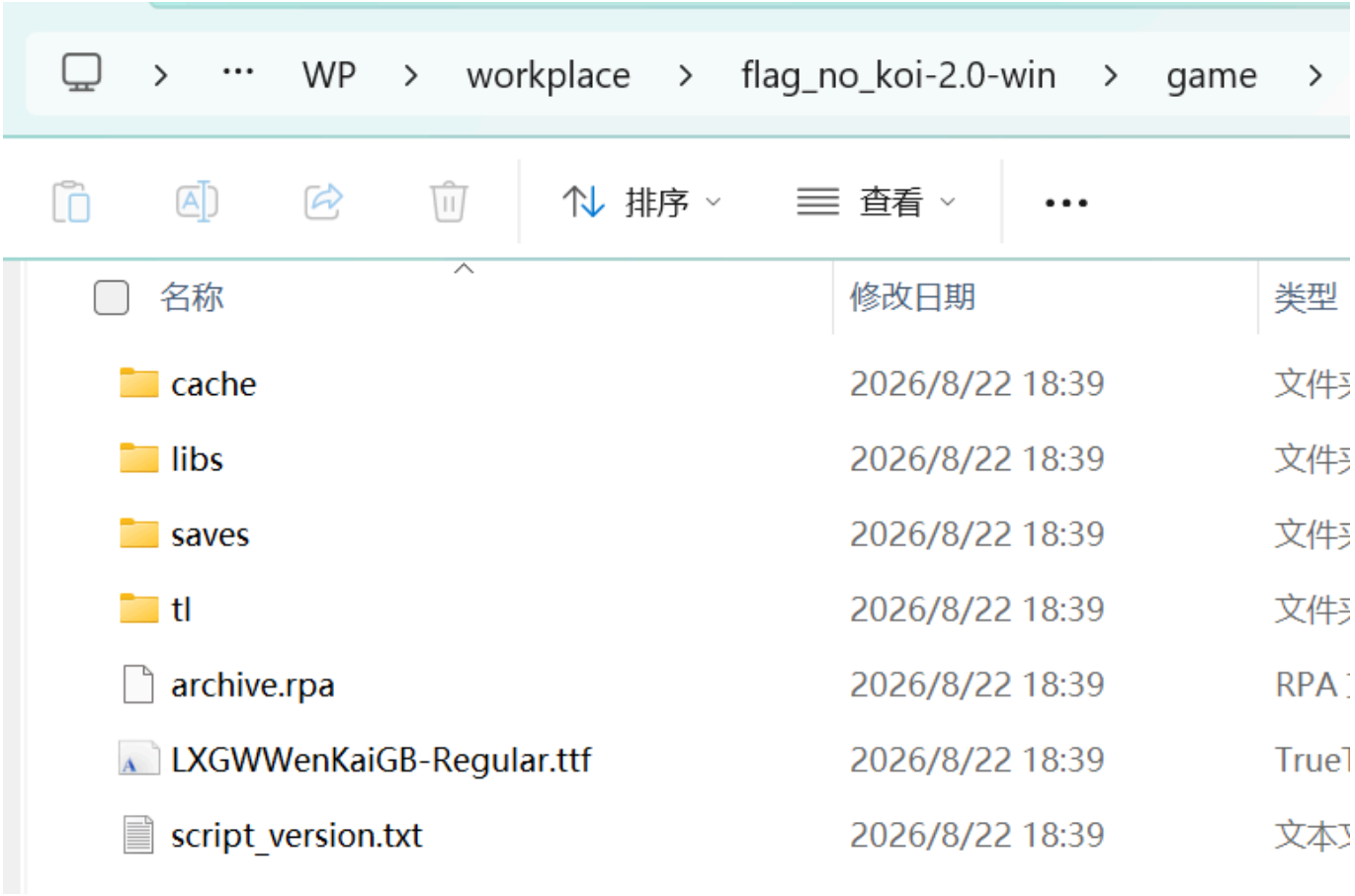
至此，我们得到了本题的主角：旮旯给木



Ren'Py解包部分

这个文件结构一眼使用Ren'Py编写，很好解包，下面开始解包

进到game文件夹内，发现一个archive.rpa，这个是renpy的自定义文件格式，可以使用unrpa项目解包




```
unrpa archive.rpa
```

```
D:\PiYuanZhouLv\sl\experiment\フラグの恋(flag no koi)\WP\workplace\flag_no_koi-2.0-win\game>unrpa archive.rpa
Extracting files from archive.rpa.

Auto-detection of the version for this archived failed—it is likely this archive is a version not supported. Try updatin
g unrpa, or submitting a request for support at https://github.com/Lattyware/unrpa/issues/new?template=new-archive-versi
on.mdHeader: “Role Play game is Awesome format version-3.0 000000000008a01d 47414c21”
You can try using --force to force a specific version rather than relying on auto-detection.
```

这里出错了，很明显，这里的存档格式被魔改了，但我们只要能找到一个offset和一个key。就可以继续使用这个工具提取

我们进到renpy文件夹下找到loader.py文件，这个文件负责解码各版本的rpa文件，可以找到所需两个值的存储位置

loader.py

```python
class RPAv3ArchiveHandler(object):
    """
    Archive handler handling RPAv3 archives.
    """

    archive_extension = ".rpa"

    @staticmethod
    def get_supported_extensions():
        return [".rpa"]

    @staticmethod
    def get_supported_headers():
        return [b"Role Play game is Awesome format version-3.0 "]

    @staticmethod
    def read_index(infile):
        l = infile.read(70)
        offset = int(l[45:61], 16)
        key = int(l[62:70], 16)
        infile.seek(offset)
        index = loads(zlib.decompress(infile.read()))

        def start_to_bytes(s):
            if not s:
                return b""

            if not isinstance(s, bytes):
                s = s.encode("latin-1")

            return s

        # Deobfuscate the index.

        for k in index.keys():
            if len(index[k][0]) == 2:
```

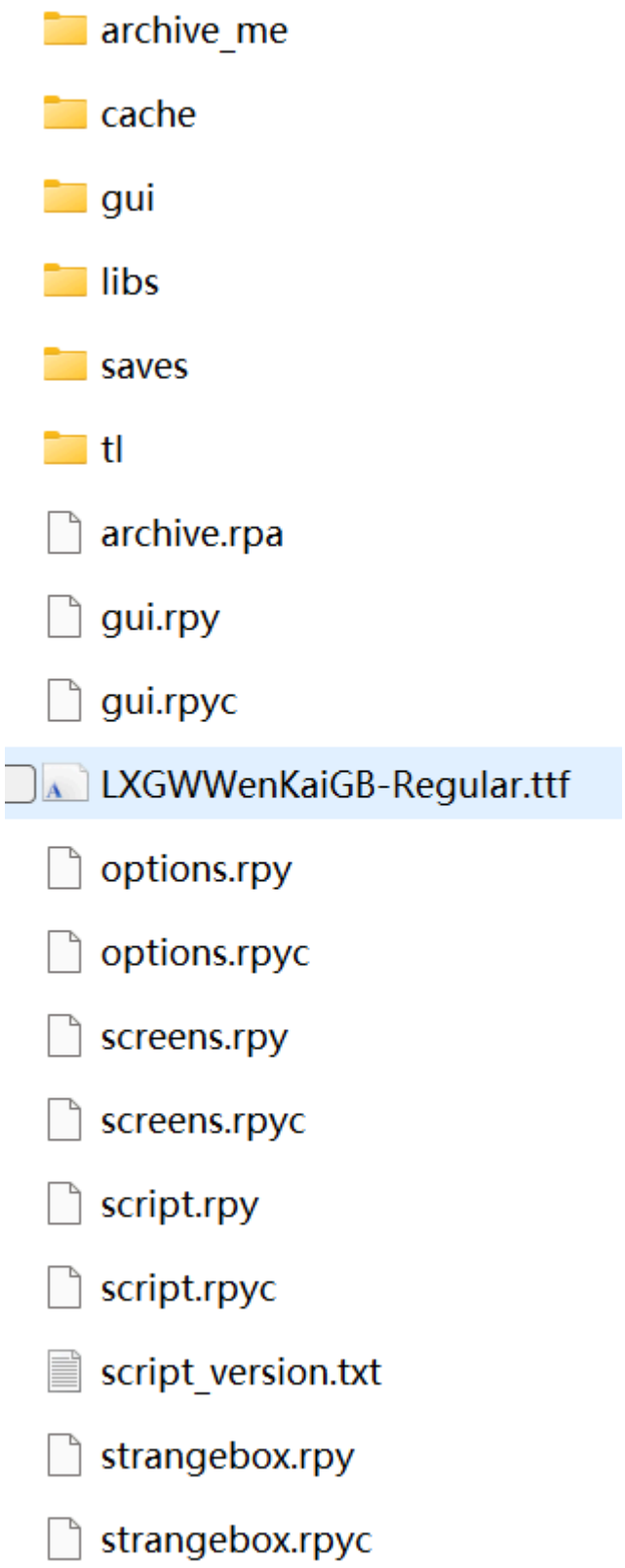
```
unrpa archive.rpa --offset 565277 --key 1195461665 --force RPA-3.0
```

这里就解包好了，当然，都看到了源码，根据源码写一个解包脚本也是可以的

另外rpa文件的zlib几乎是形同虚设，binwalk都能出来不少文件，而且key所保护的加密内容竟然是文件名而不是内容，感觉实在是有点过于草台了（x

接下来就可以使用unrpyc恢复项目源码

```
python D:\path-to\unrpyc-master\unrpyc.py .
```



核心逻辑在strangebox.rpy中，首先导入了一个模块strangebox，然后要求输入目标值target，接下来提供SMC key解锁代码，最后验证输入是否正确

接下来就是寻找SMC key和target

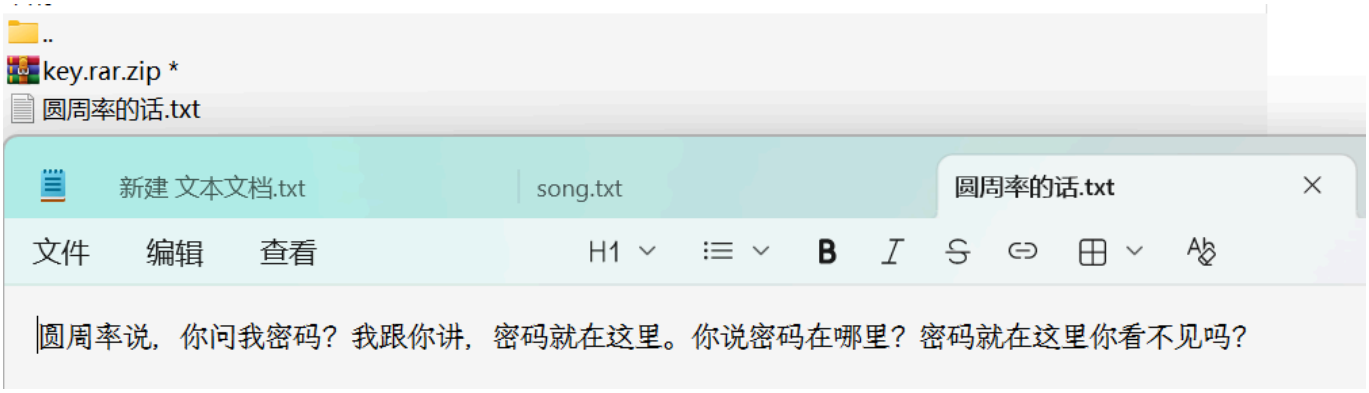
解密SMC key

注意到有一个archive_me文件夹很可疑，里面有一个piz.yek，很容易想到是倒序的zip文件，如果处于某种原因没有解析出文件名，那么就只能通过瞪眼法，注意到最后的40 30 B4 05（ZIP文件头50 4B 03 04的倒序）

接下来反转这个文件的字节

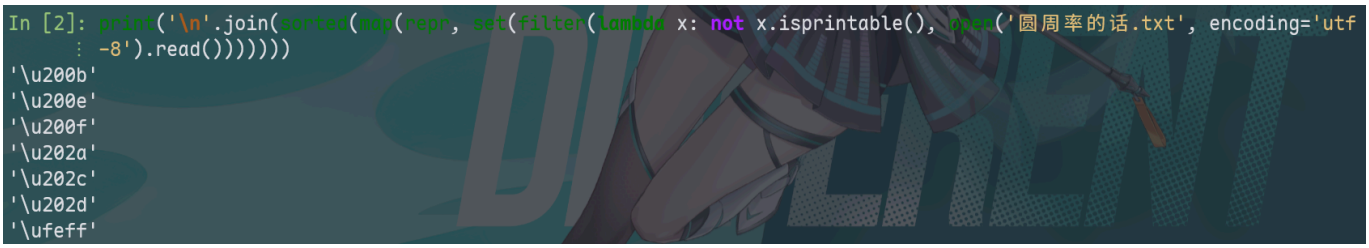
```
open('key.zip', 'wb').write(bytes.fromhex(open('piz.yek', 'rb').read().hex()[::-1]))
```

打开，是一个部分加密的zip文件，其中的文本文档很有可能含有密码，打开看看



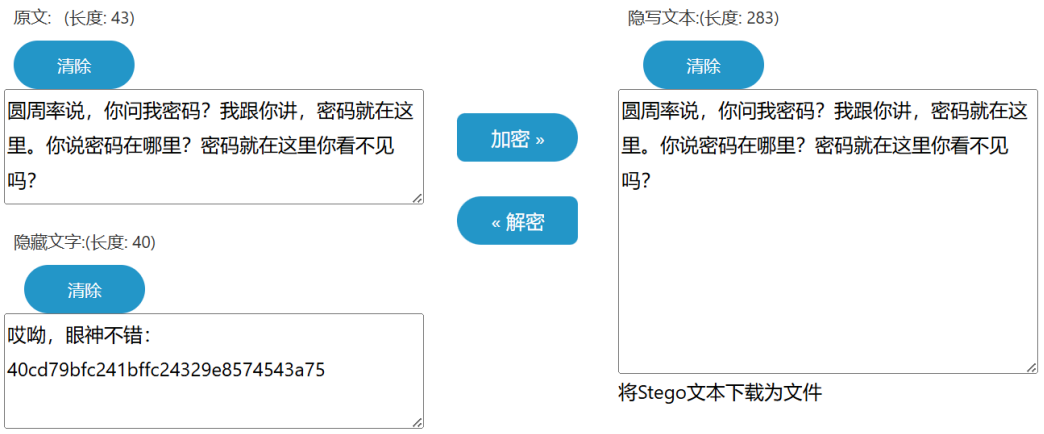
这个文件下方显示其有283个字符，而图中的字很明显没有那么多，加上文字中强调“你看不见吗”，很容易想到这是零宽字符隐写，解压文本文档，观察其中的零宽字符种类

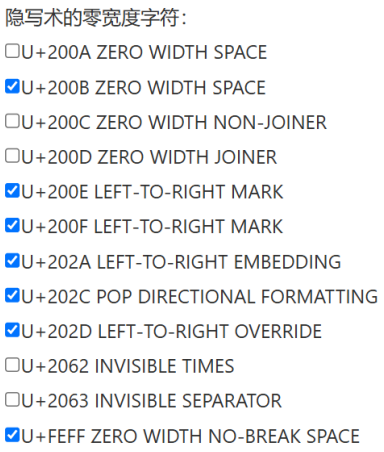
```
print('\n'.join(sorted(map(repr, set(filter(lambda x: not x.isprintable(), open('圆周率的话.txt', encoding='utf-8').read()))))))
```



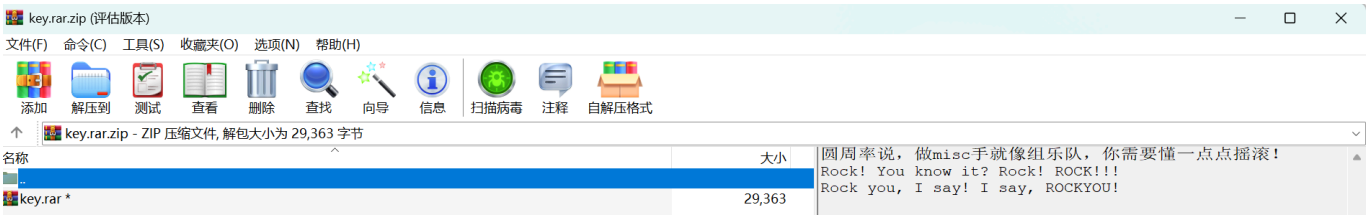
找一个零宽字符隐写网站，勾选对应零宽字符进行解密

文本隐写术示例中的文本





得到解压密码，进入下一步



注释中强调了rockyou，这是CTF中常见的一个弱密码表，其中包含超过一千四百万条密码记录。所以下一步就是利用rockyou爆破弱密码

首先使用zip2john提取哈希

```
zip2john key.rar.zip > hash1
```

删除哈希首尾多余的部分，对照哈希示例表找到模式，接下来就可以使用hashcat爆破（john其实也可以，但是可能不支持GPU加速，会慢很多）

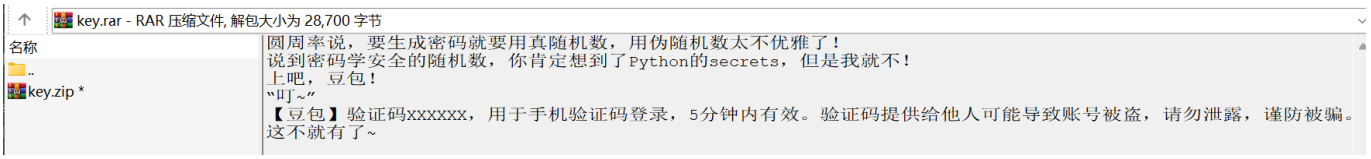
```
hashcat -a 0 -m 13600 D:\path\to\hash1 D:\path\to\rockyou.txt
```

```
3ad12849998f2b04b03dc995a58d72be9f3b52e1184ce955bad870f752aee5a031dd9e7e4e4706951ff30c81064a17b42dbad5cc091f0cf14aa7cc87
d1d59408928e1c1850f67b287a6510fdd4fbeb4b47f015fe19cc7fa8eb88003068e801e3bd89404d45e0288657153d386c5255*0b2cbe354b1a343ea
c4e*$/zip2$:aya&yassin1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13600 (WinZip)
Hash.Target......: $zip2$*0*3*0*a4d73d425fcbc1ee0ba662a7cefd77a7*cb10*.../zip2$
Time.Started.....: Sat Aug 22 19:23:40 2026 (24 secs)
Time.Estimated...: Sat Aug 22 19:24:04 2026 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.......: File (D:\PiYuanZhouLv\rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#01........:   293.0 kH/s (9.11ms) @ Accel:3 Loops:999 Thr:512 Vec:1
Speed.#03........:   115.1 kH/s (10.32ms) @ Accel:3 Loops:999 Thr:256 Vec:1
Speed.#*.........:   408.2 kH/s
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 9910272/14344385 (69.09%)
Rejected.........: 0/9910272 (0.00%)
Restore.Point....: 9851904/14344385 (68.68%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-999
Restore.Sub.#03..: Salt:0 Amplifier:0-1 Iteration:0-999
Candidate.Engine.: Device Generator
Candidates.#01...: b48596 -> avonallday1
Candidates.#03...: avonalarms -> av77563
Hardware.Mon.#01.: Temp: 60c Util: 17% Core: 600MHz Mem: 810MHz Bus:8
Hardware.Mon.#03.: N/A

Started: Sat Aug 22 19:23:36 2026
Stopped: Sat Aug 22 19:24:05 2026
```

运行24s后得到密码aya&yassin1，进入下一层



很明显，提示我们按掩码爆破六位纯数字密码，我们如法炮制，使用rar2john获得密码后使用hashcat爆破

```
└─$ rar2john key.rar
key.rar:$rar5$16$32a1ed568779dcbeeb9235022233e08c$15$420a947222c5eb5968888330d96fa7f2$8$51d35d2f50e5808e
```

这次的哈希短一些，就可以直接通过命令行调用， ~~长的指令会使咖喱味的Terminal出现一点点bug~~

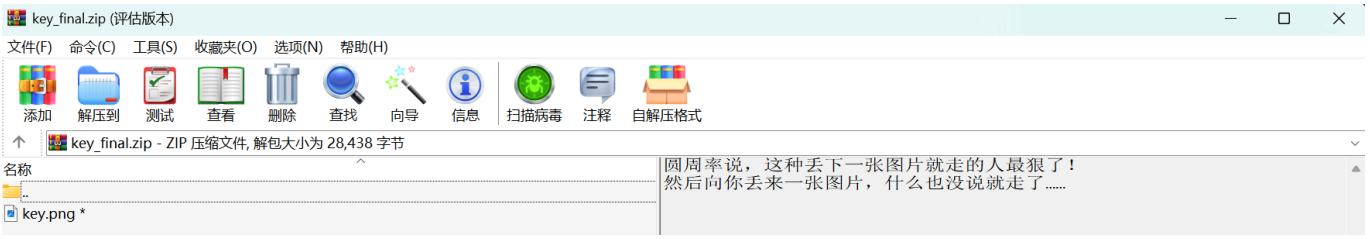
```
hashcat -a 3 -m 13000
$rar5$16$32a1ed568779dcbeeb9235022233e08c$15$420a947222c5eb5968888330d96fa7f2$8$51d35d2f50
e5808e ?d?d?d?d?d?d
```

```
$rar5$16$32a1ed568779dcbeeb9235022233e08c$15$420a947222c5eb5968888330d96fa7f2$8$51d35d2f50e5808e:833934

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13000 (RAR5)
Hash.Target......: $rar5$16$32a1ed568779dcbeeb9235022233e08c$15$420a94...e5808e
Time.Started.....: Sat Aug 22 19:32:08 2026 (10 secs)
Time.Estimated...: Sat Aug 22 19:32:18 2026 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Mask.......: ?d?d?d?d?d?d [6]
Guess.Queue......: 1/1 (100.00%)
Speed.#01........:    52297 H/s (5.54ms) @ Accel:4 Loops:512 Thr:384 Vec:1
Speed.#03........:     5680 H/s (4.54ms) @ Accel:6 Loops:128 Thr:512 Vec:1
Speed.#*.........:    57976 H/s
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 593696/1000000 (59.37%)
Rejected.........: 0/593696 (0.00%)
Restore.Point....: 0/100000 (0.00%)
Restore.Sub.#01..: Salt:0 Amplifier:7-8 Iteration:32768-32799
Restore.Sub.#03..: Salt:0 Amplifier:8-9 Iteration:8192-8320
Candidate.Engine.: Device Generator
Candidates.#01...: 891240 → 832094
Candidates.#03...: 743942 → 780151
Hardware.Mon.#01.: Temp: 78c Util: 98% Core:2760MHz Mem:9001MHz Bus:8
Hardware.Mon.#03.: N/A

Started: Sat Aug 22 19:32:04 2026
Stopped: Sat Aug 22 19:32:19 2026
```

运行10s后得到了密码833934，进入下一层



这回几乎没有提示，但是图像就让人想起明文攻击，看一下加密和存储方式

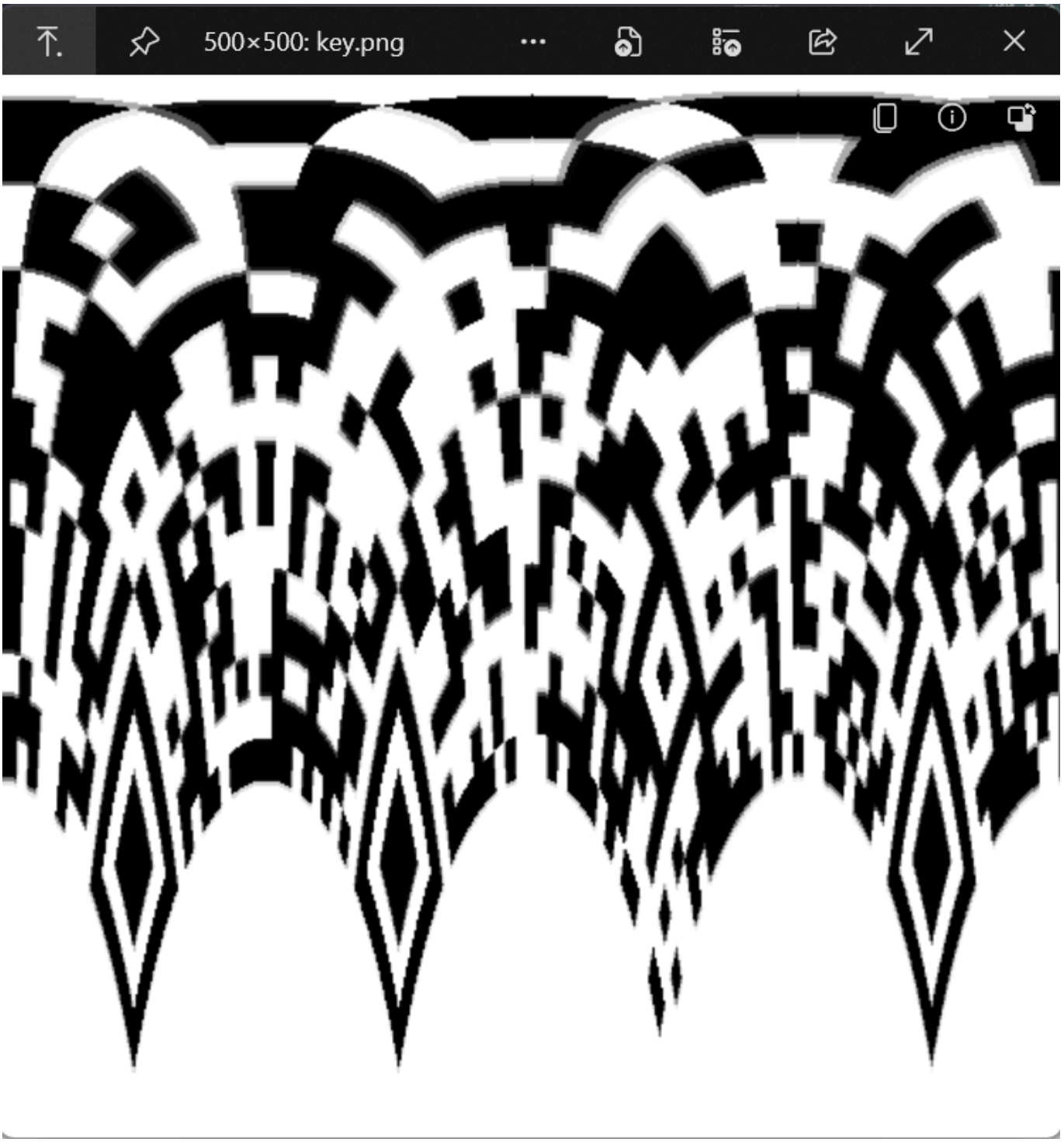


使用的是传统ZipCrypto+Store，可以利用PNG文件头进行明文攻击，提取PNG文件头，然后使用bkcrack进行明文攻击，这里的代码把破解和提取写在一起了，就不用跑两次指令了

```
bkcrack -C "D:\path\to\key_final.zip" -c key.png -p pngheader.bin -D cracked.zip
```

```
D:\PiYuanZhouLv\bkcrack-1.8.0-win64>bkcrack -C "D:\PiYuanZhouLv\sl\experiment\フラグの恋(flag no koi)\WP\workplace\flag_no_koi-2.0-win\game\archive_me\key_final.zip" -c key.png -p pngheader.bin -D cracked.zip
bkcrack 1.8.0 - 2025-08-18
[19:37:36] Z reduction using 9 bytes of known plaintext
100.0 % (9 / 9)
[19:37:37] Attack on 721203 Z values at index 6
Keys: e003da05 0c187819 7efde0a4
56.1 % (404952 / 721203)
Found a solution. Stopping.
You may resume the attack with the option: --continue-attack 404952
[19:40:08] Keys
e003da05 0c187819 7efde0a4
[19:40:08] Writing decrypted archive cracked.zip
100.0 % (1 / 1)
```

解压得到key.png



这玩意一眼二维码，但是是什么变换呢？这里致敬（？）了一下DDLC中的`natsuki.chr`，采取的是极坐标变换，只要做一次Rect2Polar就好了



扫出来就是SMC key了


there_1s_@_5tran9e_6ox_aNd_heRe_i5_7he_keY_6ut_vvha7s_!
n_it??!
已复制
重新扫描

提取target

既然target也要输入，那么target在哪里呢？压缩包里还留下了一个存档文件，而renpy会自动存储所有的变量值，回去再看看target的代码：

```
label image_input:
    $ target = renpy.input(prompt="输入幻像……")

    "你输入的幻像是：\n[target]"

    $ _history_list[-1].what = (lambda x: (lambda y, z: f'{y}: {glitch(:)}')(*
    $ glitch_last_input()
    $ renpy.block_rollback()
```

由此可见，我们可以从存档文件里读取target的值

最简单的办法就是使用开发者模式加载存档，然后在控制台中输出target的值

为此，我们先将存档复制到renpy的存档文件夹：%APPDATA%/Renpy/[project-name]-[project-id]/，然后在game文件夹下创建一个rpy文件，写入

```
define config.developer = True
```

接下来启动游戏按Shift+O就可以……

新建 文本文档.txt | song.txt | traceback.txt

文件 编辑 查看 H1

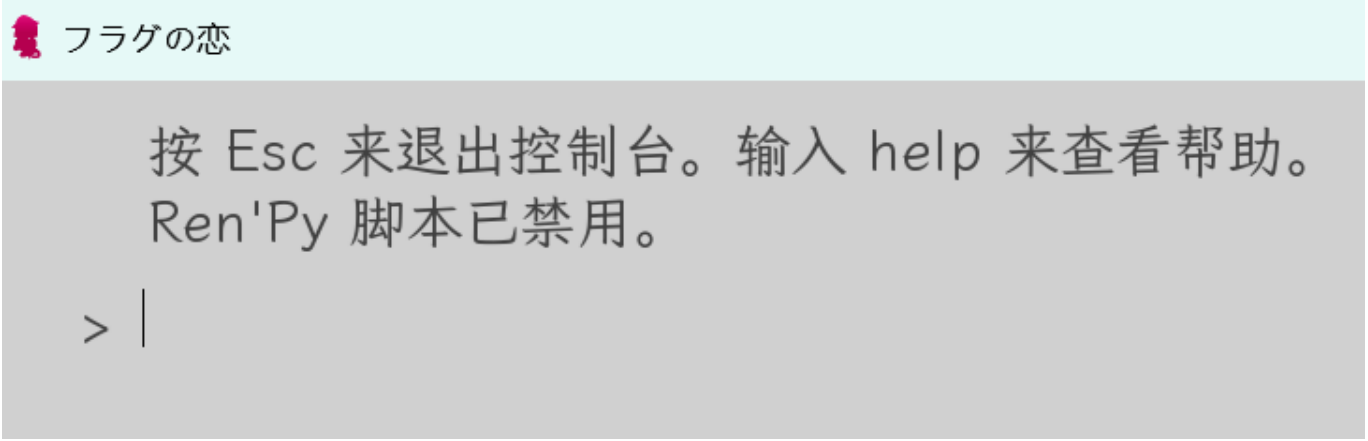
```
I'm sorry, but an uncaught exception occurred.

While running game code:
  File "renpy/display/render.pyx", line 494, in renpy.display.render.render_screen
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 320, in renpy.display.render.render_for_size
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
  File "renpy/display/render.pyx", line 170, in renpy.display.render.render
  File "renpy/display/render.pyx", line 267, in renpy.display.render.render
Exception: Could not find font 'SourceHanSansLite.ttf'.

During handling of the above exception, another exception occurred:
```

炸了，提示找不到字体，最简单的办法就是把文件夹里的字体复制一份改个名字就行

接下来启动游戏按Shift+O就可以打开控制台了



然后读取存档，启动控制台，输入target就有target的值了~

フラグの恋

```
按 Esc 来退出控制台。输入 help 来查看帮助。
Ren'Py 脚本已禁用。
Ren'Py 脚本已启用。
target
'SYEvg6pw9bmftakduXHUMLsK89oIpkbzH3YHdGnbCi2svAMf7hviAVszOCmiPvxILVt2fALAI3I3I2ZOffDo++YQiHYMr995'
> |
```

这里是可以自由执行Ren'Py/Python指令的，所以可以把target写到文件里，方便复制

另外各位的AI肯定是选择手撕save文件的，下面也分析一下

首先，可以注意到save文件的文件头是504B0304，所以改为zip并解压，里面几个文件明显没有用，唯一有用的是log，这是一个pickle文件

我让D老师写了一个在无对应类时进行polly fill的脚本，最后输出为json，附在最后

输出来的内容开头就送“store.target”……欸，怎么不一样?


```
[
  {
    "store.target": "BgwQIzJjJSJsNzgzITQ+MSANHQAYGSYebWw6OSU+Ny8dZgwdMRI7NxY8ZyYjFBgzYj0jPBQDJi8aFjg8BSMtORkDIWczFBkUHGY5ZhxnDxozMxE6fn4MBDwdDBgnbGxg",
    "store._menu": false,
    "store._window_auto": true,
    "store._last_raw_what": "游戏结束了~",
    "store._last_say_what": "游戏结束了~",
    "store._last_say_kwargs": {
      "interact": true
    },
    "store.quick_menu": true,
    "store._history": true,
    "store._version": "2.0",
    "store._history_list": [
      {
        "__unknown__": {
          "module": "renpy.character",
          "name": "HistoryEntry",
```


这里就需要看一下renpy文件夹下的loadsave.py了，可以看到save/load逻辑中有我加上去的一段逻辑

```
def save(slotname, extra_info="", mutate_flag=False, include_screenshot=True, extra_json=None):

    if mutate_flag:
        renpy.revertable.mutate_flag = False

    roots = renpy.game.log.freeze(None)

    if 'store.target' in roots:
        import base64
        roots['store.target'] = base64.b64encode(bytes(i^0x55 for i in roots['store.target'].encode())).decode()

    if renpy.config.save_dump:
        dump_paths("save_dump.txt", **{"renpy.game.log": renpy.game.log}, **roots)
```

```
def load(filename):

    renpy.session["traceback_load"] = "_traceback" in json

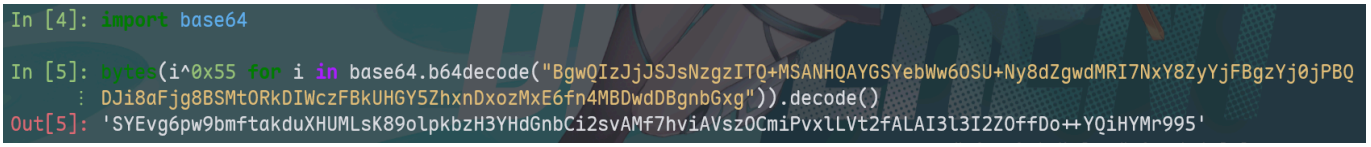
    renpy.exports.call_in_new_context("_before_load")

    roots, log = loads(log_data)

    if 'store.target' in roots:
        import base64
        roots['store.target'] = bytes(i^0x55 for i in base64.b64decode(roots['store.target'])).decode()

    log.unfreeze(roots, label="_after_load")
```

这里对存储的target进行了简单加/解密，但是加密后的格式与加密前一致，都是base64，所以一下子还真不一定看得出来



因为这个操作对于游戏是透明的，所以通过游戏提取就不会被阴到，突发奇想想到了之后就立刻加上了

~~Galgame有旮旯给木的做法，你应该手动解包，然后手动输入魔改的偏移，最后使用Ren'Py相关的知识解决问题，什么叫你丢给AI就出来了，旮旯给木不是这样的，我不接受！~~ 😤

分析strangebox

接下来就是最神秘的存在：strangebox

通过搜索在lib/python3.12文件夹下找到了strangebox.pyd

逆向分析一下，发现不是很好逆向的样子……

喂喂，这是Misc题，你想到哪去了，怎么会在逆向上卡你呢？所以有没有什么不用逆向的办法呢？

有的有的，兄弟有的

我们可以把它作为一个黑盒分析，一旦这个算法有些许设计上的缺陷，就算混淆得再复杂，照样能解出

接下来我们要导入这个模块，由于pyd与python版本相关，所以要启动lib/py3-windows-x86_64下的python，或者直接在控制台中测试

```
import strangebox
strangebox.unlock('there_1s_@_5tran9e_6ox_aNd_heRe_i5_7he_keY_6ut_vvha7s_!n_it??!')
import base64
for i in range(16):
    print(f"{'1'*-~i:<16} {base64.b64decode(strangebox.E(b'1'*-~i)).hex()}")
```



我们发现，虽然加密结果整体上没什么规律，但是它有一个致命缺点：距离到一定程度后，后面的数据将不会影响前面加密的结果，也就是说，我们可以分组对其进行爆破，这样减少了非常大的枚举空间，最后的爆破成本是可以接受的（但是还是比较久，稍安勿躁）

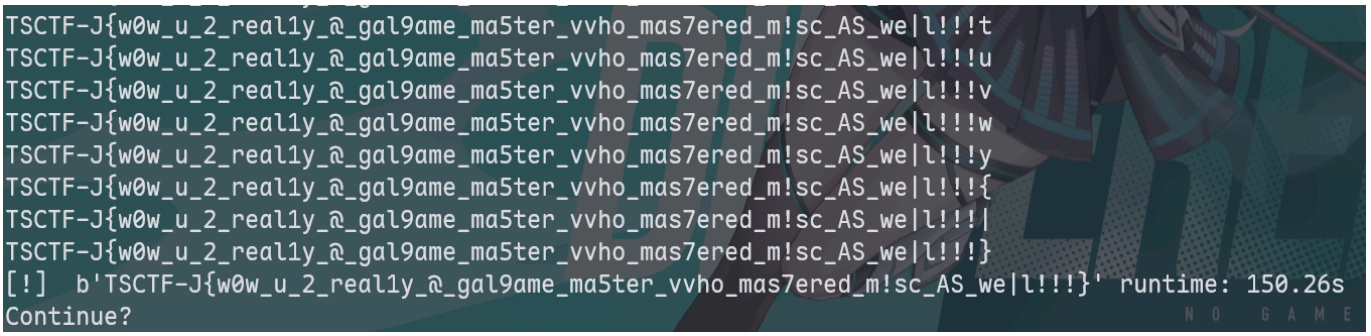
下面给出爆破脚本

```
from strangebox import *
unlock('there_1s_@_5tran9e_6ox_aNd_heRe_i5_7he_keY_6ut_vvha7s_!n_it??!')
import base64
target =
base64.b64decode(b'SYEvg6pw9bmftakduXHUMLsK89olpkbzH3YHdGnbCi2svAMf7hviAVszOCmiPvxlLVt2fAL
AI3l3I2ZOffDo++YQiHYMr995')

CHARSET = '!012579@AJS_acdeghlmorstuvwy{|}'.encode()
import itertools
def guess(target, prefix=b''):
    for l in range(0 if (last := (len(target)-3 == len(prefix))) else 3, 3 if last else
4):
        for g in itertools.product(CHARSET, repeat=l):
            g = bytes(g)
            print((prefix+g).decode())
            if target.startswith(base64.b64decode(E(prefix+g))[:len(prefix)+2 if not last
else None]):
                if last:
                    yield g
                else:
                    for gn in guess(target, prefix+g):
```

```
                        yield g+gn

import time
st = time.time()
for g in guess(target, b'TSCTF-'):
    print('[!] ', b'TSCTF-'+g, f'runtime: {time.time()-st:.2f}s')
    if not input("Continue?"):
        break
```



~~耗时两分半~~ 拿到flag

完结撒花~

出题人视角

首先这是一道misc大杂烩，因为我看着misc里面考的都比较灵活，基础没考什么，所以就往里面额外塞了一大堆misc基础知识。

首先是旮旯给木基础 & renpy解包/target提取，是承接宣讲内容主题的一部分，使用的都是我在实际调查时遇到的套路，而这些基本都是我们misc玩剩了的，考察搜索能力、面对未知知识的主动学习和灵活应变能力。

解密SMC key考察了含密码的压缩包的常见考法，但含密码的压缩包还远远不止这点知识点，比如伪加密、以文件名作为密码之类的考法都没有囊括在内。最后的图像极坐标变换也是不常考的点，更常见的变换应该是ZigZag来着，可以去了解一下ZigZag变换后的二维码特征。

为了给RenPy内置的python编译pyd文件，我可是费了不少功夫，通过询问D老师，最后解决了。简单来说就是配置UCRT编译环境，由dll文件生成符号文件，从不记得什么地方凑齐了includes，最后链接编译，有机会会在博客展开讲

由于要SMC，所以没办法直接Cython，最后是让D老师把代码用CPython API重写了一遍，为了尽可能防止直接逆向解决，最后开了O3+strip，不过感觉AI还能能分析出来，说不定有更高效的解密方式呢

顺带一题，由于本题计划考察黑盒逻辑分析，所以我写的算法根本没有考虑可逆性，会有一些映射到同一结果的，碰撞率在13%左右，不过比我大笔一挥几面A4纸的初版算法好多了（碰撞率好像是80%左右，一堆多解）。

附：D老师给的解包pickle的脚本

注意：请不要解包任何安全性未知的pickle文件！能RCE的！

```
import pickle
import json
```

```
import sys
from datetime import datetime, date, time
from decimal import Decimal
from collections.abc import Mapping, Sequence

# ---------- 通用占位类 ----------
class Unknown:
    """
    当原始类缺失时，用此占位类替代。
    支持列表操作（append, extend, 整数索引）和字典风格操作（字符串键），
    同时保留所有属性（通过 __dict__）。
    """
    def __new__(cls, *args, **kwargs):
        instance = super().__new__(cls)
        instance._module = getattr(cls, '_module', 'unknown')
        instance._name = getattr(cls, '_name', 'unknown')
        instance._list_items = []          # 存储列表元素
        return instance

    def __init__(self, *args, **kwargs):
        pass

    def __setstate__(self, state):
        # 如果状态是字典，更新到 __dict__
        if isinstance(state, dict):
            self.__dict__.update(state)
        else:
            # 其他状态保存备用
            self._state = state

    # ---------- 列表支持 ----------
    def append(self, item):
        self._list_items.append(item)

    def extend(self, iterable):
        self._list_items.extend(iterable)

    def __len__(self):
        return len(self._list_items)

    def __iter__(self):
        return iter(self._list_items)

    # ---------- 索引支持（兼容列表和字典） ----------
    def __getitem__(self, key):
        if isinstance(key, int):
            return self._list_items[key]
        # 字符串键作为属性访问（优先从 __dict__ 获取）
        return self.__dict__.get(key)

    def __setitem__(self, key, value):
        if isinstance(key, int):
            self._list_items[key] = value
        else:
```

```
            self.__dict__[key] = value

    def __repr__(self):
        return f"<Unknown {self._module}.{self._name} (len={len(self._list_items)})>"


class SafeUnpickler(pickle.Unpickler):
    def find_class(self, module, name):
        try:
            return super().find_class(module, name)
        except (AttributeError, ImportError):
            # 动态创建 Unknown 子类并记录原类信息
            class UnknownSubclass(Unknown):
                _module = module
                _name = name
            return UnknownSubclass


# ---------- 递归转换器 ----------
def to_jsonable(obj):
    """将任意 Python 对象转换为 JSON 可序列化的结构"""
    # 基础类型
    if obj is None or isinstance(obj, (str, int, float, bool)):
        return obj
    if isinstance(obj, bytes):
        return repr(obj)
    if isinstance(obj, (datetime, date, time)):
        return obj.isoformat()
    if isinstance(obj, Decimal):
        return float(obj) if obj.is_finite() else str(obj)

    # 优先处理 Unknown 占位对象（包含列表元素 + 额外属性）
    if isinstance(obj, Unknown):
        # 列表元素（递归转换）
        list_items = [to_jsonable(item) for item in obj._list_items]
        # 额外属性（排除内部字段）
        attrs = {}
        for key, value in obj.__dict__.items():
            if key not in ('_module', '_name', '_list_items', '_state'):
                attrs[key] = to_jsonable(value)
        # 将 _state 也纳入（如果有）
        if hasattr(obj, '_state') and obj._state is not None:
            attrs['_state'] = to_jsonable(obj._state)
        return {
            '__unknown__': {
                'module': obj._module,
                'name': obj._name,
                'list': list_items,      # 列表数据（若是列表子类）
                'attrs': attrs           # 其他属性
            }
        }

    # 序列（列表、元组等）
    if isinstance(obj, Sequence) and not isinstance(obj, str):
```

```
        return [to_jsonable(item) for item in obj]

    # 映射（字典、有序字典等）
    if isinstance(obj, Mapping):
        return {str(k): to_jsonable(v) for k, v in obj.items()}

    # 集合
    if isinstance(obj, (set, frozenset)):
        return [to_jsonable(item) for item in obj]

    # 其他自定义对象：尝试提取 __dict__
    if hasattr(obj, '__dict__'):
        return to_jsonable(vars(obj))

    # 最终降级为 repr
    return repr(obj)


def load_pickle_to_json(filepath, indent=2):
    try:
        with open(filepath, 'rb') as f:
            unpickler = SafeUnpickler(f)
            obj = unpickler.load()
    except FileNotFoundError:
        raise FileNotFoundError(f"文件不存在：{filepath}")
    except pickle.UnpicklingError as e:
        raise ValueError(f"解析 Pickle 失败：{e}")

    jsonable = to_jsonable(obj)
    return json.dumps(jsonable, indent=indent, ensure_ascii=False)


if __name__ == '__main__':
    # 强制标准输出使用 UTF-8，避免重定向编码问题
    if hasattr(sys.stdout, 'reconfigure'):
        sys.stdout.reconfigure(encoding='utf-8')
    else:
        import io
        sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

    if len(sys.argv) < 2:
        print("用法：python unpickle.py <pickle文件路径>")
        sys.exit(1)

    pickle_path = sys.argv[1]
    try:
        result = load_pickle_to_json(pickle_path)
        print(result)
    except Exception as e:
        print(f"错误：{e}", file=sys.stderr)
        sys.exit(1)
```