hgame2024 week2

Crypto

奇怪的图片plus

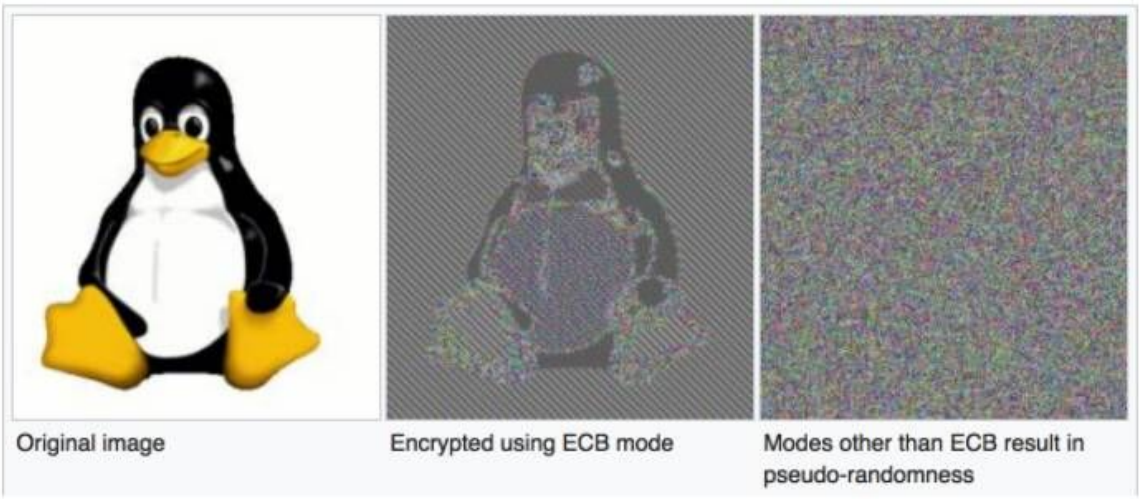
把这俩图片传上去就能得到key

gitf:8693346e81fa05d8817fd2550455cdf6

原理

缺点

1. 同样的明文块会被加密成相同的密文块，不会隐藏明文分组的统计规律。正如下图所示



为了解决统一明文产生相同密文的问题，提出了其它的加密模式。

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Util.number import long_to_bytes,bytes_to_long
```

```
import os
from PIL import Image, ImageFont, ImageDraw
import struct
import random


def image_to_bytes(image):
    width, height = image.size
    pixel_bytes = []
    for y in range(height):
        for x in range(width):
            pixel = image.getpixel((x, y))
            pixel_bytes.extend(struct.pack('BBB', *pixel))
    image_bytes = bytes(pixel_bytes)
    return image_bytes


def bytes_to_image(image_bytes, width, height):
    pixel_bytes = list(image_bytes)
    reconstructed_image = Image.new('RGB', (width, height))
    for y in range(height):
        for x in range(width):
            start = (y * width + x) * 3
            pixel = struct.unpack('BBB', bytes(pixel_bytes[start:start + 3]))
            reconstructed_image.putpixel((x, y), pixel)
    return reconstructed_image


def xor_images(image1, image2):
    if image1.size != image2.size:
        raise ValueError("Images must have the same dimensions")
    xor_image = Image.new("RGB", image1.size)
    pixels1 = image1.load()
    pixels2 = image2.load()
    xor_pixels = xor_image.load()
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2, g1 ^ g2, b1 ^ b2)
    return xor_image


def draw_text(image, width, height, token):
    font_size = 20
    font = ImageFont.truetype("arial.ttf", font_size)
    text_color = (255, 255, 255)
    x = 0
    y = (height // 2) - 10
    draw = ImageDraw.Draw(image)
    draw.text((x, y), token, font=font, fill=text_color)
    pixels = image.load()
    for x in range(width):
        for y in range(height):
            if pixels[x, y] != (0, 0, 0):
```

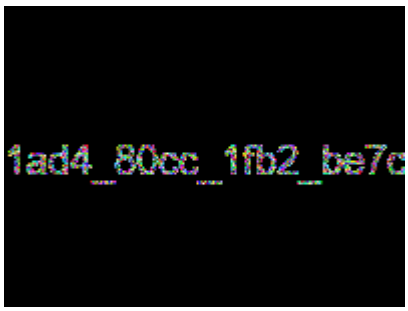
```
                pixels[x, y] = (random.randint(0, 255), random.randint(0, 255),
random.randint(0, 255))
    return image


image1=Image.open("C:/Users/13945/Desktop/Crypto/hgame24/week2/strange_image_plus
/encrypted_flag.png")
key = long_to_bytes(0x8693346e81fa05d8817fd2550455cdf6)
m = image_to_bytes(image1)
iv=long_to_bytes(bytes_to_long(m[0:16])^0)
F = AES.new(key=key, mode=AES.MODE_ECB)
iv = F.decrypt(iv)

G = AES.new(key=key,mode=AES.MODE_OFB,iv=iv)
n = pad(m, F.block_size)
n = G.encrypt(n)
image2 = bytes_to_image(n, 200, 150)
image2.show()
```

解密先求iv再解密

具体可以看OFB模式的原理



hgame{1ad4_80cc_1fb2_be7c}

midRSA

非预期

```
from Crypto.Util.number import *
m0=13292147408567087351580732082961640130543313742210409432471625281702327748963274496942276607
print(long_to_bytes(m0))
```

b'hgame{0ther_cas3s_0f_c0ppr3smith}\xff\xff\xff\xff\xff'

midRSA revenge

```
#sage

from Crypto.Util.number import *
```

```
n=27814334728135671995890378154778822687713875269624843122353458059697288888640572922486287556431241786461159513236128914176680497775619694684903498070577307810263677280294114135929708745988406963307279767028969515305895207028282193547356414827419008393701158467818535109517213088920890236300281646288761697842280633285355376389468360033584102258243058885174812018295460196515483819254913183079496947309574392848378504246991546781252139861876509894476420525317251695953355755164789878602945615879965709871975770823484418665634050103852564819575756950047691205355599004786541600213204423145854859214897431430282333052121
c=45622131411586708863820720303449463624470661111162172357784872909606923006795813266301862566144713150175868450263938320833284468193969812445918857181352714977229246413953073671761974170494592607563206407212536151643563112184575318655929799335527077981805770297378339158985115911402931029655170145674869891423134483518791755930544026956061332689320474812799925490210291960537036388958113672416409687957317387028080662045408746697035899865473675525702322507814701853710 1
m0=99999002810033577734203106811693308232665325338039056 37

#print(len(bin(n))-2)  2048
#print(len(bin(m0))-2) 183

R.<x> = PolynomialRing(Zmod(n), implementation='NTL')
m = m0*2^128 + x
M = m((m^e - c).small_roots()[0])
print(long_to_bytes(int(M)))
```

coppersmith求 n**(1/e)以内的根

显然符合条件

b'hgame{c0ppr3smith_St3re0typed_m3ssag3s}'

backpack

非预期

```
from Crypto.Util.number import *
enc=87111417256785349029747857011344936698879376017284464400756682491335008814816294996881254121833 9
print(long_to_bytes(enc))
```

b'hgame{M@ster_0f ba3kpack_m4nag3ment!}\x00\x0e#'

backpack revenge

```
#sage

from Crypto.Util.number import *
import hashlib
```

```
a=[747630795102616991263455259 79, 5172504947006895081047848750 7,
47190309269514609005045330671, 649559896406501398183482149 27,
685599372386236236191140659 17, 723113391701121854014968670 01,
708173360642547816402733540 39, 705381088265397857743616053 09,
437825309424818656212933810 23, 582343281865780362910570662 37,
688082712654788585701269169 49, 616602004709381538360454838 87,
632707269818515446203592313 07, 429047764866976916696399292 29,
415456372017875316374276033 39, 740128390556498913971728708 91,
569437947956412606749536768 27, 517373919021877591880786874 53,
492643689995616599861828839 07, 600442212373871040545978619 73,
638470463502605207610436878 17, 621281466995821807790139835 61,
651093134232128526479302999 81, 668256358698317310926840393 51,
677632651477912720837807523 27, 611678440839991796697026016 47,
551160159278687568590079619 43, 523444885180556720822803775 51,
523758778919423123200318039 19, 696590359415641192916404047 91,
525632820851786467678143828 89, 568106273122864204941091920 29,
497558777990068890638825665 49, 438589016724517567544748451 93,
679237436151549832911456245 23, 516894555147285474239951626 37,
674801311517071556725275833 21, 593962122483305800721846480 71,
634105288752204897994752492 07, 480114092885508802292805781 49,
625619692603911329568182859 37, 448261586642837794103306159 71,
704462187599762399477511620 51, 565098473798366000335019425 37,
501542879711798313550684431 53, 490605071160958611749714671 49,
542368482942996246321605210 71, 641866264289749761084671968 69]
bag=120254819682601389900652731 4947

L = matrix(ZZ,49,49)
for i in range(48):
    L[i,i]=2
    L[i,48]=a[i]
    L[48,i]=1
L[-1,-1]=bag

L=L.LLL()
l=[0]*48
for i in range(48):
    if L[0][i]==1:
        l[i]=1

l=l[::-1]

p=0
for i in range(48):
    p=p*2+l[i]
flag='hgame{'+hashlib.sha256(str(p).encode()).hexdigest()+'}'
print(flag)
```

格

hgame{04b1d0b0fb805a70cda94348ec5a33f900d4fd5e9c45e765161c434fa0a49991}

babyRSA

```
#sage

from Crypto.Util.number import *

p=14213355454944773291
q=6184356205162070038634855117537193048606497844115920076561833974376400103329 7
c=10500213872246694649593663865603821400004347575163902508525511396508874927246190689258661625026492234819249659798645278628115115643622957406519396542284 1
gift=9751789326354522940

d0=inverse(0x10001,p-1)
e=pow(gift,d0,p)-114514
#print(e) 73561

res = mod(c, p4*q).nth_root(e, all = true)

for i in res:
    flag=long_to_bytes(int(i))
    if b'hgame' in flag:
        print(flag)
        break
```

数论求e

然后AMM开根

看了半天AMM发现sage里能直接开

b'hgame{Ad1eman_Mand3r_Mi11er_M3th0d}'