Κρυπτογραφία- Απαλλακτική Εργασία

CRYPTOHACK

Κοντοπούλου Δέσποινα

Πανεπιστήμιο Πειραιώς- Τμήμα Πληροφορικής

15/01/2024

Σύνδεσμος profile στο CryptoHack:
https://cryptohack.org/user/thespianexe
Σύνδεσμος repository στο Github:
https://github.com/Thespiann/CryptoHackDoc

Σύνολο πόντων εώς 15/1/2024:

- RSA Starter 6 (25 pts)
- Diffie-Hellman Starter 5 (40 pts)
- Modes of Operation Starter (15 pts)

  960pts μετρήσιμοι στην εργασία

CHALLENGES

INTRODUCTION

Finding Flags

Εισαγωγή flag σε form.

crypto{y0ur_f1rst_fl4g}

Great Snakes

Εκτέλεση συννημένου.

crypto{z3n_0f_pyth0n}

Network Attacks

For this challenge, connect to `socket.cryptohack.org` on port 11112. Send a JSON object with the key `buy` and value `flag`.

Επίλυση

```
from pwn import *
import json

HOST = "socket.cryptohack.org"
PORT = 11112

r = remote(HOST, PORT)
```

```python
def json_recv():
   line = r.readline()
   return json.loads(line.decode())
def json_send(hsh):
   request = json.dumps(hsh).encode()
   r.sendline(request)

print(r.readline())
print(r.readline())
print(r.readline())
print(r.readline())

request = {
   "buy": "flag"
}
json_send(request)
response = json_recv()
print(response)
```

Εξήγηση επίλυσης

1. Χρήση βιβλιοθήκης pwn για σύνδεση στον διακομιστή.
2. Αποστολή αντικειμένου json με json_send()
3. Ανάγνωση απαντησης με json_recv()
4. Εκτύπωση απαντήσεων

`crypto{sh0pp1ng_f0r_fl4g5}`

GENERAL

ENCODING

ASCII

Using the below integer array, convert the numbers to their corresponding ASCII characters to obtain a flag.

Επίλυση

```
flag = "".join([chr(i) for i in [99, 114, 121, 112, 116,
111, 123, 65, 83, 67, 73, 73, 95, 112, 114, 49, 110, 116,
52, 98, 108, 51, 125]])
print(flag)
```

Εξήγηση επίλυσης

Το flag είναι η ένωση (join) των αντίστοιχων ASCII χαρακτήρων των ακεραίων(chr(int)).

crypto{ASCII_pr1nt4bl3}

HEX

Included below is a flag encoded as a hex string. Decode this back into bytes to get the flag.

Επίλυση

```
hex='63727970746f7b596f755f77696c6c5f62655f776f726b696e67
5f776974685f6865785f737472696e67735f615f6c6f747d'
print(bytes.fromhex(hex))
```

crypto{You_will_be_working_with_hex_
strings_a_lot}

BASE 64

Take the below hex string, *decode* it into bytes and then *encode* it into Base64.

Επίλυση

```
import base64
hex='72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7
bf'
bytes=bytes.fromhex(hex)
print(base64.b64encode(bytes).decode())
```

crypto/Base+64+Encoding+is+Web+Safe/

BYTES AND BIG INTEGERS

Convert the following integer back into a message:

Επίλυση

```
from Crypto.Util.number import bytes_to_long,
long_to_bytes
int_value =
11515195063862318899931685488813747395775516287289682636
499965282714637259206269 #given int
byte_data =long_to_bytes(int_value) #convert int to bytes
ascii_bytes=bytes(byte_data) #convert bytes to ascii
print(ascii_bytes.decode('utf-8')) #print ascii
```

crypto{3nc0d1n6_4ll_7h3_w4y_d0wn}

ENCODING CHALLENGE

Now you've got the hang of the various encodings you'll be encountering, let's have a look at automating it. Can you pass all 100 levels to get the flag? Connect at `socket.cryptohack.org 13377`

Επίλυση

```
import base64
import codecs
import json
from pwn import *
import Crypto.Util.number

HOST = "socket.cryptohack.org"
PORT = 13377
r= remote(HOST, PORT)
#functions for reading lines and handling json objects
def readline():
   return r.readuntil(b"\n")
def json_recv():
   line = readline()
   return json.loads(line.decode())
def json_send(hsh):
   request = json.dumps(hsh).encode()
   r.write(request)

for i in range(100):
   received = json_recv()#receive data
   encoding = received['type']
   encoded = received['encoded']
#decode using the specified encoding
   if encoding == "base64":
       decoded = base64.b64decode(encoded).decode()
   elif encoding == "hex":
       decoded = bytes.fromhex(encoded).decode()
   elif encoding == "rot13":
       decoded = codecs.decode(encoded, "rot13")
   elif encoding == "bigint":
       decoded =
Crypto.Util.number.long_to_bytes(int(encoded,
16)).decode()
   elif encoding == "utf-8":
       decoded = "".join(chr(o) for o in encoded)
#prints
   print(f"{i + 1}) {encoding}:")
   print(f"{encoded} ---> {decoded}\n")
#send decoded back to get the flag
```

```
    json_send(
        {"decoded": decoded})
flag = readline()
flag = json.loads(flag.decode())
print("\nFlag:")
print(str(flag["flag"]))
```

Εξήγηση Επίλυσης

Το πρόγραμμα συνδέεται στον διακομιστή και την θύρα που δίνονται και έπειτα λαμβάνει 100 κρυπτογραφημένα json objects οπου το κάθε ένα εχει τα πεδία type και encoded. Αποκωδικοποιεί τα μηνύματα, στέλνει πίσω το decoded και έπειτα διαβάζει και εκτυπώνει το flag.

crypto{3nc0d3_d3c0d3_3nc0d3}

XOR

XOR STARTER

Given the string `label`, XOR each character with the integer 13. Convert these integers back to a string and submit the flag as `crypto{new_string}`.

Επίλυση

```
xor= [chr(ord(char)^13) for char in "label"]
new=''.join(xor)
print("crypto{"+new+"}")
```

crypto{aloha}

XOR PROPERTIES

Below is a series of outputs where three random keys have been XOR'd together and with the flag. Use the above properties to undo the encryption in the final line to obtain the flag.

Επίλυση

```
k1 =
bytes.fromhex('a6c8b6733c9b22de7bc0253266a3867df55acd
e8635e19c73313')
k2k1 =
bytes.fromhex('37dcb292030faa90d07eec17e3b1c6d8daf94c
35d4c9191a5e1e')
k2k3 =
bytes.fromhex('c1545756687e7573db23aa1c3452a098b71a7f
bf0fddddde5fc1')
flagk1k2k3=
bytes.fromhex('04ee9855208a2cd59091d04767ae47963170d1
660df7f56f5faf')

def xor(bytes1,bytes2):
   return bytes([b1 ^ b2 for b1, b2 in
zip(bytes1,bytes2)])

k2= xor(k2k1,k1)
k3=xor(k2k3,k2)
flag=xor(xor(flagk1k2k3,k1),xor(k2,k3))
print(flag)
```

crypto{x0r_i5_ass0c1at1v3}

FAVOURITE BYTE

I've hidden some data using XOR with a single byte, but that byte is a secret. Don't forget to decode from hex first.

Επίλυση

```
from pwn import xor
hex='73626960647f6b206821204f21254f7d694f7624662065622127
234f726927756d'
bytes=bytes.fromhex(hex)
for i in range(255):
   flag= xor(bytes,i)
   if flag.startswith(b'crypto'):
       print(xor(bytes,i))
       print("secret key is: " + str(i))
       break
```

Εξήγηση επίλυσης

Το πρόγραμμα δοκιμάζει τις δυνατές τιμές ενός byte i ως κλειδί στην xor με το δοθέν hex. Εαν το αποτέλεσμα ξεκινά άπο crypto, εκτυπώνει το αποκρυπτογραφημένο κείμενο και το κλειδι.

`crypto{0x10_15_my_f4v0ur173_by7e}`

YOU EITHER KNOW, XOR YOU DON'T

I've encrypted the flag with my secret key, you'll never be able to guess it.

Επίλυση

```
from pwn import xor
hex='0e0b213f26041e480b26217f27342e175d0e070a3c5b103e
2526217f27342e175d0e077e263451150104'
bytes=bytes.fromhex(hex)
crypto='crypto{'.encode()
#print(xor(bytes,crypto)) :
#b'myXORke+y_Q\x0bHOMe$~seG8bGURN\x04DFWg)a|\x1dTM!an
#\x7f'
key='myXORkey'.encode()
print(xor(bytes,key))
```

Εξήγηση επίλυσης

Κάνοντας xor με το format των flag, προκύπτει το

```
b'myXORke+y_Q\x0bHOMe$~seG8bGURN\x04DFWg)a|\x1dTM!an\x7f'
```

αρα μαντέυουμε πως το κλειδί ειναι `myXORkey`.

`crypto{1f_y0u_Kn0w_En0uGH_y0u_Kn0w_1t_4
11}`

LEMUR XOR

I've hidden two cool images by XOR with the same secret key so you can't see them!

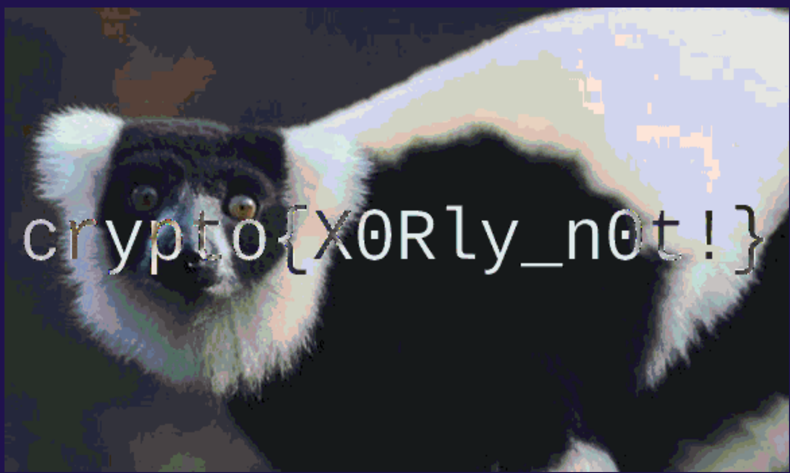
Επίλυση

```
from PIL import Image
import numpy as np
def image_to_array(image_path):#for getting each pictures pixels
   img = Image.open(image_path)
   return np.array(img)
def array_to_image(array, output_path):#for turning the pixels into images
   img = Image.fromarray(array)
   img.save(output_path)
flag_path = "flag.png"
lemur_path = "lemur.png"
flag_array = image_to_array(flag_path)
lemur_array = image_to_array(lemur_path)
new_array = np.bitwise_xor(flag_array, lemur_array) #xor-ing the pixels of
the images
output_path = "new_flag.png" #save in new file
array_to_image(new_array, output_path)
```

Εξήγηση Επίλυσης

Το πρόγραμμα μετατρέπει τις φωτογραφίες σε arrays με pixel με την συνάρτηση image_to_array(). Έπειτα εφαρμόζεται xor στα arrays και μετατρέπεται το array που προέκυψε σε .jpg με την array_to_image().

new_flag.jpg:



crypto{X0Rly_n0t!}

MATHEMATICS

GREATEST COMMON DIVISOR

Now calculate `gcd(a,b)` for `a = 66528, b = 52920`

Επίλυση

```
def gcd(a,b):#function for getting the greatest common divisor
    if a==0:
        return b #if a is 0 then b is the gcd
    else:
        return gcd(b%a,a) #recursion, we call gcd() with the updated values
print(gcd(66528,52920))
```

```
1512
```

EXTENDED GCD

Using the two primes `p = 26513, q = 32321`, find the integers `u,v` such that `p * u + q * v = gcd(p,q)`. Enter whichever of `u` and `v` is the lower number as the flag.

Επίλυση

```
p = 26513
q = 32321
#p * u + q * v = gcd(p,q)
def egcd(a,b):
    if a==0:
        return b,0,1
    else:
        gcd, u, v= egcd(b%a,a)
        return gcd, v-(b//a)*u, u
gcd,u,v=egcd(p,q)
print(min(u,v))
```

Εξήγηση Επίλυσης:
Υλοποιειται ο επαυξημένος αλγόριθμος του Ευκλείδη για τον υπολογισμό του gcd. Συγκεκριμένα:

- *αν a=0 -> μεγιστος κοινος διαιρέτης b*
- *αν a!= 0-> egcd(b mod a, a)*
  - *-return gcd, u, v -(b//a)*u, u*

εκτυπώνω το ελάχιστο απο τα u,v

MODULAR ARITHMETIC 1

Calculate the following integers:

$11 \equiv x \bmod 6$

$8146798528947 \equiv y \bmod 17$

Επίλυση

```
print(min(11%6,8146798528947%17))
```

4

MODULAR ARITHMETIC 2

Now take the prime $p = 65537$. Calculate $273246787654^{65536} \bmod 65537$.

Επίλυση

```
p=65537
a=273246787654
#fermats little theorem
#if p=prime, a^p mod p= a
#if a%p!=0 (a not divisble by p), then a^(p-1)-1 =
x*p, so a^(p-1)=1 mod p
if a%p !=0:
   print(1)
```

1

MODULAR INVERTING

What is the inverse element: 3 * d ≡ 1 mod 13?

Επίλυση

```
#For all elements g in the field, there exists a unique integer d such that g * d ≡ 1 mod p
d=0
while (3*d)%13 !=1:
   d=d+1
print(d)
```

9

DATA FORMATS

PRIVACY-ENHANCED MAIL

Extract the private key *d* as a decimal integer from this PEM-formatted RSA key.

Επίλυση

```
from Crypto.PublicKey import RSA
with open('privacy_enhanced_mail.pem', 'r') as key_file:
   key = RSA.importKey(key_file.read())
   private_key_d_decimal = key.d
   print(private_key_d_decimal)
```

15682700288056331364787171045819973654991149949197959929860861228180021707316851924456205543665565810892674190059831330231436970914474774562714945620519144389785158908994181951348846017432506464163564960993784254153395406799101314760033445065193429592512349952020982932218524462341002102063435489318813316464511621736943938440710470694912336237680219746204595128959161800595216366237538296447335375818871952520026993102148328897083547184286493241191505953601668858941129790966909236941127851370202421135897091086763569884760099112291072056970636380417349019579768748054760104838790424708988260443926906673795975104689

CERTAINLY NOT

Find the modulus of the certificate, giving your answer as a decimal.

Επίλυση

```
from Crypto.PublicKey import RSA
from cryptography import x509
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend

with open('2048b-rsa-example-cert.der', 'rb') as der_file:
   cert = x509.load_der_x509_certificate(der_file.read(),
default_backend())

with open('2048b-rsa-example-cert.pem', 'wb') as pem_file:

pem_file.write(cert.public_bytes(encoding=serialization.Encoding.PEM))
with open('2048b-rsa-example-cert.pem', 'r') as key_file:
   key = RSA.import_key(key_file.read())
   print(type(key))
   print(key.n)
```

Εξήγηση Επίλυσης

Το πρόγραμμα χρησιμοποιει το cryptography module για την φόρτωση του der πιστοποιητικού cert. Επειτα το μετατρέπει σε μορφή PEM με την χρήση της

```
cert.public_bytes(encoding=serialization.Encoding.PEM)
```

και χρησιμοποιεί την RSA για να φορτώσει το κλειδι σε δεκαδικό.

22825373692019530804306212864609512775374171823993708516509897631547513634635856375624003737068034549047677999310941837454378829351398302382629658264078775456838626207507725494030600516872852306191255492926495965536379271875310457319107936020730050476235278671528265817571433919561175665096171189758406136453987966255236963782666066962654678464950075923060327358691356632908606498231755963567382339010985222623205586923466405809217426670333410014429905146941652293366212903733630083016398810887356019977409467374742266276267137547021576874204809506045914964491063393800499167416471949021995447722415959979785959569497

SSH KEYS

Extract the modulus *n* as a decimal integer from Bruce's SSH public key.

Επίλυση

```
from Crypto.PublicKey import RSA
pub = open("bruce_rsa.pub").read()
key = RSA.importKey(pub)
print(key.n)
```

```
393140627292252344843619459982009301624147265815180155284509451857950781599060
045966925960364526153292761115298494284088989875653206089485704517530014576580
063349900545173887208138126700406986555739563855004111420614308540360723410929
328633639355275689398460521435298870525863897945473651499731422366907590078380
671539888031069594594514775513291903797388907519178597779786155722867815953888
215354471779710040109643506235947412975562545383188249060356013447704323543320
270894861523453698471587211334381276010281232318039154449603016365304693141472
385137455487303658228238990483859766828654333742658168081779603871122840144324
465516219930235201796499786667731716101408311673053587552128663185810276896109
885120940097389939396493160506785600541099863184267303090107800840864961353814
379995980368504156696451448980921196298453432234839442801090898431894041169896
115073120431667064667697636195882852822983761079584314504824349290
9
```

TRANSPARENCY

Find the subdomain of cryptohack.org which uses these parameters in its TLS certificate, and visit that subdomain to obtain the flag.

Επίλυση

```
import requests, json, sys
target = 'cryptohack.org'
req =
requests.get("https://crt.sh/?q=%.{d}&output=json".fo
rmat(d=target))
json_data = json.loads(req.text)
for (index, certificate) in enumerate(json_data):
    url = str(certificate['name_value'])
```

```python
    if url.endswith(target):
        try:
            response_req = requests.get('https://' +
url).text

            if (response_req.startswith('crypto')):
                print('flag : ' + response_req)
                break
        except:
            continue
```

Επεξήγηση επίλυσης

Με την βοήθεια του ιστοτόπου crt.sh γίνεται αναζήτηση για πιστοποιητικά που να περιέχουν 'cryptohack.org' με ένα http request.

Περιηγούμαστε στην λίστα απο τα πιστοποιητικά που επιστράφηκε με το json και για κάθε ένα:

1. Ελέγχουμε αν το url λήγει σε 'cryptohack.org'
2. Αν ναι, κάνουμε ξανα http request στον ιστοτοπο.
3. Αν η απάντηση ξεκινά απο 'crypto' εκτυπώνω το flag και γίνεται break.

`crypto{thx_redpwn_for_inspiration}`

MATHEMATICS

MODULAR MATH

QUADRATIC RESIDUES

Find the quadratic residue and then calculate its square root. Of the two possible roots, submit the smaller one as the flag.

Επίλυση

```
p = 29
ints = [14, 6, 11]
#for i in range(29):
#  print(pow(i,2,29))
#  if pow(i,2,29)==18:
#    print("i="+str(i))

for i in ints:
 for a in range(29):
   if pow(a,2,29)==i:
     print("a="+str(a))
```


LEGENDRE SYMBOL

Given the following 1024 bit prime and 10 integers, find the quadratic residue and then calculate its square root; the square root is your flag. Of the two possible roots, submit the larger one as your answer.

Επίλυση

```
p = 1015...
ints = [25081841...]
```

```python
def legendre(a, p):
    legendre=pow(a,(p-1)//2,p)
    if(legendre==p-1):
        legendre=-1
    return legendre

for i in ints:
    l= legendre(i,p)
    if l==1:
        sqrt= pow(i,(p+1)//4,p)
        print(sqrt)
```

Εξήγηση επίλυσης.

Αρχικά υλοποιείται συνάρτηση για τον υπολογισμό του Legendre, ωστε να βρεθεί ποιος απο τους αριθμούς είναι quadratic residue. Ελέγχονται οι ακέραιοι, και στην περίπτωση που βρεθει το quadratic residue υπολογίζουμε την τετραγωνική του ρίζα, υψώνοντας τον ακέραιο εις την (p+1)//4 με mod p.

93291799125366706806545638475797430512104976066103610269938025709952247020061090804870186195285998727680200979853848718589126765742550855954805290253592144209552123062161458584575060939481368210688629862036958857604707468372384278049741369153506182660264876115428251983455344219194133033177700490981696141526

MODULAR SQUARE ROOT

Find the square root of a modulo the 2048-bit prime p. Give the smaller of the two roots as your answer.

Επίλυση (σε περιβάλλον sageMath)

```
from sage.rings.finite_rings.integer_mod import
square_root_mod_prime
a=8479994...
p=305318...
square_root_mod_prime(mod(a, p))
```

236233930768304863832777329858048929893213750552050038833827105205373474
786235177964731417681795335907187156004112528991924714607490715161276264
086819962118655952206833803260099131188222401602122267224313936218046123
264673246584884042545825793088785658337960096776173859678287785131848935
567982281315512304570528511209944814642675511016000251559241885043210364
181581107154845628426350780558944507365756538185052136796967569976075531
078462357707644003774768176030243492493211364006173877760119462224419275
802418085391624442725406544196255728257284916277274079898964794864520734
97374574454404050571568975083685319391200

SYMMETRIC CIPHERS

HOW AES WORKS

KEYED PERMUTATIONS

What is the mathematical term for a one-to-one correspondence?

crypto{bijection}

RESISTING BRUTEFORCE

What is the name for the best single-key attack against AES?

crypto{biclique}

STRUCTURE OF AES

Included is a `bytes2matrix` function for converting our initial plaintext block into a state matrix. Write a `matrix2bytes` function to turn that matrix back into bytes, and submit the resulting plaintext as the flag.

Επίλυση

```
def bytes2matrix(text):
    """ Converts a 16-byte array into a 4x4 matrix.  """
    return [list(text[i:i + 4]) for i in range(0, len(text), 4)]
def matrix2bytes(matrix):
    """ Converts a 4x4 matrix into a 16-byte array.  """
    bytes_array = bytearray()
    for i in matrix:
        for j in i:
            bytes_array.append(j)
    return bytes_array
matrix = [
    [99, 114, 121, 112],
    [116, 111, 123, 105],
    [110, 109, 97, 116],
    [114, 105, 120, 125],
]
print(matrix2bytes(matrix))
```

crypto{inmatrix}

ROUND KEYS

Complete the `add_round_key` function, then use the `matrix2bytes` function to get your next flag.

Επίλυση

```
from StructureOfAES import matrix2bytes
state = [
   [206, 243, 61, 34],
   [171, 11, 93, 31],
   [16, 200, 91, 108],
   [150, 3, 194, 51],
]
round_key = [
   [173, 129, 68, 82],
   [223, 100, 38, 109],
   [32, 189, 53, 8],
   [253, 48, 187, 78],
]
def add_round_key(s, k):
   added_round_key = [
       [scolumn ^ kcolumn for scolumn, kcolumn in zip(scol, kcol)]
       for scol, kcol in zip(s, k)
   ]
   return added_round_key

print("The added round key is: \n ", add_round_key(state,
round_key))
print("The byte array is: \n ", matrix2bytes(add_round_key(state,
round_key)))
```

crypto{r0undk3y}

CONFUSION THROUGH SUBSTITUTION

Implement `sub_bytes`, send the state matrix through the inverse S-box and then convert it to bytes to get the flag.

Επίλυση

```
import base64
from StructureOfAES import matrix2bytes
from RoundKeys import add_round_key
s_box = (
   0x63,...)
inv_s_box = (
   0x52,...)
state = [
   [251, 64, 182, 81],
   [146, 168, 33, 80],
   [199, 159, 195, 24],
   [64, 80, 182, 255],
]
def sub_bytes(s, sbox=s_box):
   sub_bytes =""
   for i in range(4):
       for j in range(4):
           sub_bytes+=chr(sbox[s[i][j]])
   return sub_bytes
print(sub_bytes(state, sbox=inv_s_box))
```

Εξήγηση επίλυσης

Ορίζεται η κενή αρχικα συμβολοσειρα sub bytes. Επειτα υλοποιειται η διαδικασία της αντικαταστασης καθε byte του πίνακα s με το sbox και ο χαρακτήρας που προκύπτει προστίθεται στην συμβολοσειρά.

crypto{l1n34rly}

DIFFUSION THROUGH PERMUTATION

We've provided code to perform MixColumns and the forward ShiftRows operation. After implementing `inv_shift_rows`, take the state, run `inv_mix_columns` on it, then `inv_shift_rows`, convert to bytes and you will have your flag.

Επίλυση

```
def shift_rows(s):
   s[0][1], s[1][1], s[2][1], s[3][1] = s[1][1], s[2][1], s[3][1],
s[0][1]
   s[0][2], s[1][2], s[2][2], s[3][2] = s[2][2], s[3][2], s[0][2],
s[1][2]
   s[0][3], s[1][3], s[2][3], s[3][3] = s[3][3], s[0][3], s[1][3],
s[2][3]
def inv_shift_rows(s):
   s[1][1], s[2][1], s[3][1], s[0][1] = s[0][1], s[1][1], s[2][1],
s[3][1]
   s[2][2], s[3][2], s[0][2], s[1][2] = s[0][2], s[1][2], s[2][2],
s[3][2]
   s[3][3], s[0][3], s[1][3], s[2][3] = s[0][3], s[1][3], s[2][3],
s[3][3]
# learned from http://cs.ucsb.edu/~koc/cs178/projects/JT/aes.c
xtime = lambda a: (((a << 1) ^ 0x1B) & 0xFF) if (a & 0x80) else (a
<< 1)
def mix_single_column(a):
   # see Sec 4.1.2 in The Design of Rijndael
   t = a[0] ^ a[1] ^ a[2] ^ a[3]
   u = a[0]
   a[0] ^= t ^ xtime(a[0] ^ a[1])
   a[1] ^= t ^ xtime(a[1] ^ a[2])
   a[2] ^= t ^ xtime(a[2] ^ a[3])
   a[3] ^= t ^ xtime(a[3] ^ u)
def mix_columns(s):
   for i in range(4):
       mix_single_column(s[i])
def inv_mix_columns(s):
   # see Sec 4.1.3 in The Design of Rijndael
```

```
    for i in range(4):
        u = xtime(xtime(s[i][0] ^ s[i][2]))
        v = xtime(xtime(s[i][1] ^ s[i][3]))
        s[i][0] ^= u
        s[i][1] ^= v
        s[i][2] ^= u
        s[i][3] ^= v

    mix_columns(s)
state = [
    [108, 106, 71, 86],
    [96, 62, 38, 72],
    [42, 184, 92, 209],
    [94, 79, 8, 54],
]
inv_mix_columns(state)
inv_shift_rows(state)

for i in range(4):
    for j in range(4):
        print(chr(state[i][j]), end="")
```

Εξήγηση Επίλυσης

Δημιουργείται η συνάρτηση inv_shift_rows(s) με βαση την δοθεν shift_rows() ωστε να αντιστρέφονται οι αλλαγές που προκάλεσε η δεύτερη στο s. Έπειτα ο τελικός πίνακας εκτυπώνεται σε χαρακτήρες ASCII για την ευρεση του flag.

`crypto{d1ffUs3R}`

BRINGING IT ALL TOGETHER

Copy in all the building blocks you've coded so far, and complete the `decrypt` function that implements the steps shown in the diagram. The decrypted plaintext is the flag.

Επίλυση

```
#Only showing decrypt func
def decrypt(key, ciphertext):
   round_keys = expand_key(
       key)  # Remember to start from the last round key and
work backwards through them when decrypting

   # Convert ciphertext to state matrix
   text = bytes2matrix(ciphertext)
   # Initial add round key step
   add_round_key(text, round_keys[10])

   for i in range(N_ROUNDS - 1, 0, -1):
       inv_shift_rows(text)
       inv_sub_bytes(text)
       add_round_key(text, round_keys[i])
       inv_mix_columns(text)
   # Run final round (skips the InvMixColumns step)
   inv_shift_rows(text)
   inv_sub_bytes(text)
   add_round_key(text, round_keys[0])
   # Convert state matrix to plaintext

   plaintext = matrix2bytes(text)
   return plaintext

print(decrypt(key, ciphertext))
```

crypto{MYAES128}

*Υπάρχει η επίλυση για το *Modes of Operation Starter* στο github, όμως δεν προσμετρούνται οι πόντοι.

RSA

STARTER

RSA STARTER 1

Find the solution to $101^{17} \bmod 22663$

Επίλυση

```
print(pow(101,17,22663))
```

19906

RSA STARTER 2

"Encrypt" the number 12 using the exponent `e = 65537` and the primes `p = 17` and `q = 23`. What number do you get as the ciphertext?

Επίλυση

```
p=17
q=23
e=65537
N=p*q #calculation of N
print(pow(12,e,N))#modular exponantion with my
parameters
```

301

RSA STARTER 3

Given `N = p*q` and two primes:

p = 857504083339712752489993810777

q = 1029224947942998075080348647219

What is the totient of N?

Επίλυση

```
p = 857504083339712752489993810777
q = 1029224947942998075080348647219
N=p*q
"""Euler's totient function is a multiplicative
function, meaning that if two numbers and are
relatively prime, then, φ(p*q)=φ(p)*φ(q). If p is a
prime, then f(p)=p-1, because all the numbers up to p
are not divisible by p"""
totient=(p-1)*(q-1)
print(totient)
```

882564595536224140639625987657529300394956519977044270821168

RSA STARTER 4

Given the two primes:

p = 857504083339712752489993810777

q = 1029224947942998075080348647219

and the exponent:

e = 65537

What is the private key d?

Επίλυση

```
p = 857504083339712752489993810777
q = 1029224947942998075080348647219
e=65537
totient=(p-1)*(q-1)# (f(N)=f(p*q)= totient. if
gcd(e,f(N))=1, d=e^-1 mod totient.In the context of
RSA, it is typical to choose an encryption exponent e
that is coprime with the totient, so d is:
print(pow(e,-1,totient))
```

121832886702415731577073962957377780195
510499965398469843281

RSA STARTER 5

Use the private key that you found for these parameters in the previous challenge to decrypt this ciphertext.

Επίλυση

```
n=
882564595536224140639625987659416029426239230804614613279163
#from previous challenge
fn=88256459553622414063962598765752930039495651997704427082116
8
e = 65537
key= pow(e,-1,fn)
c =
77578995801157823671636298847186723593814843845525223303932
#decrypted=c^key mod n
decrypted=pow(c,key,n)
print(decrypted)
```

13371337

*Υπάρχει η επίλυση για το *RSA Starter 6* στο github, όμως δεν προσμετρούνται οι πόντοι.

PRIMES PART 1

FACTORING

Factorise the 150-bit number 510143758735509025530880200653196460532653147 into its two constituent primes. Give the smaller one as your answer.

Επίλυση

```
#from sympy import factorint wont run for me
#i would do
#print(min(factorint(51014375873550902553088020065319646053265
#3147))), instead
#i put the number in this calculator
#https://www.alpertron.com.ar/ECM.HTM
a = 19704762736204164635843
b = 25889363174021185185929
print(min(a,b))
```

19704762736204164635843

MONOPRIME

Why is everyone so obsessed with multiplying two primes for RSA. Why not just use one?

Επίλυση

```
from Crypto.Util.number import long_to_bytes
n = 1717313...
e = 65537
ct = 1613675...
d=pow(e,-1,n-1)# the private key is calculated, the totient of prim N is
N-1
ciphertext=pow(ct,d,n)
print(long_to_bytes(ciphertext))
```

crypto{0n3_pr1m3_41n7_pr1m3_l01}

MANYPRIME

Using one prime factor was definitely a bad idea so I'll try using over 30 instead.

Επίλυση

```
from Crypto.Util.number import inverse, long_to_bytes
import gmpy2
n = 58064...
e = 65537
ct = 3207214...
factors=[9282...]
#using http://factordb.com/index.php?query=58064239...
f=1
for x in factors:
   f *= (x-1)
#d=pow(e,-1,f-1)
d=gmpy2.invert(e,f)
print(long_to_bytes(pow(ct,d,n)))
```

crypto{700_m4ny_5m4ll_f4c70r5}

PUBLIC EXPONENT

SALTY

Smallest exponent should be fastest, right?

Επίλυση

```
from Crypto.Util.number import long_to_bytes
n = 110581...
e = 1
ct = 449812...
print(long_to_bytes(ct))# since d = 1 mod (p-1)(q-1) if
e=1
```

crypto{saltstack_fell_for_this!}

DIFFIE-HELLMAN

STARTER

DIFFIE-HELLMAN STARTER 1

Given the prime $p = 991$, and the element $g = 209$, find the inverse element d such that $g * d \bmod 991 = 1$.

Επίλυση

```
p= 991
g=209
# g*d mod p =1
for x in range(p):
   res= (g*x)%p
   if res==1:
       print(x)
       break
```

569

DIFFIE-HELLMAN STARTER 2

For the finite field with $p = 28151$ find the smallest element g which is a primitive element of F_p.

Επίλυση

```
def is_primitive_root(g, p):#Check if g is a
primitive root modulo p
   for i in range(2, p - 1):
       if pow(g, i, p) == 1:
           return False
   return True

def min_prim(p):# Find the smallest primitive element
```

```
in the finite field Fp
   for g in range(2, p):

       if is_primitive_root(g, p):
           return g

p = 28151
primitive_element = min_prim(p)
print(primitive_element)
```

7

DIFFIE-HELLMAN STARTER 3

Calculate the value of g^a mod p.

Επίλυση

```
g=2
p=24103...
a=97210...
print(pow(g,a,p))
```

```
180685769784072652332258672182091135848942012812924807867393365353
```

```
180685769784072652332258672182091135848942012812924807867393365353
393068167618175384941171571417360435232355655878375925266106118632
027421488310488605016436812919171970740229157733048549951352236828
939535952390140613802502252241242923897159127216051914467238953239
367383226507005731948539979310118268217746536439627742471754343401
766634380727697086447583039177640395755067836236831977656602511849
206219694145126563805440017724857227134254861610396741199043735792
4
```

DIFFIE-HELLMAN STARTER 4

You and Alice are now able to calculate a shared secret, which would be infeasible to calculate knowing only {g,p,A,B}. What is your shared secret?

Επίλυση

```
g=2
p=2410...
```

```
A= 7024...
b= 120...
B=518...
#person A number a, person B num b, a,b private
#g,p public
#k=g^(ab) mod p= messfromA^b mod p
print(pow(A,b,p))
```

```
1174130740413820656533832746034841985877302086316388380165984436672307692443711310285014138545204369495478725102882673427892104539120952393788961051992901649694063179853598311473820341215879965343136351436410522850717408445802043003164658348006577408558693502220285700893404674592567626297571222027902631157072143330043118418467094237965591198440803970726604537807146703763571606861448354607502654664700390453794493176794678917352634029713320615865940720837909466
```

*Υπάρχει η επίλυση για το *Diffie-Hellman Starter 5* στο github, όμως δεν προσμετρούνται οι πόντοι.

CRYPTO ON THE WEB

JSON WEB TOKENS

TOKEN APPRECIATION

To prove them wrong, decode the JWT above to find the flag. There are online tools to do this quickly, but working with Python's PyJWT library will prepare you best for future challenges.

Επίλυση

```
import jwt
encoded="eyJ0eX.."
print(jwt.decode(encoded,options={"verify_signature":
False}))#i can decode without a signature
```

crypto{jwt_contents_can_be_easily_viewed}