Cryptography

S2Day5crypto.md

by Nathan Hailu

recall

LAST TIME TOPICS

Topics

- What is Cryptography?
- Types of Cryptography
- Terms of Cryptography
- Kinds of Cryptography
- Tools
- Python for Cryptography
- Obfuscation

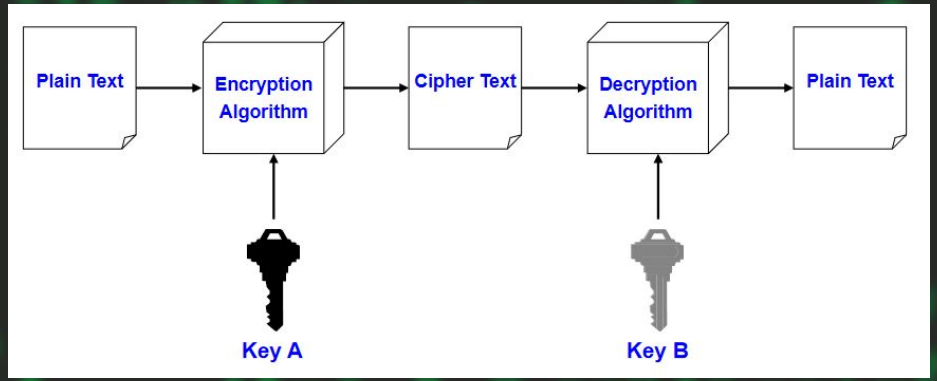
What is Cryptography?

- Cryptography is the science of secret, or hidden writing
- Crypto => Hidden/Secret | Graphy => Writing
- Used to secure your data/text.
- It has two main Components:
  a. Encryption
     i. Practice of hiding messages so that they can not be read by anyone other than the intended recipient
  b. Authentication & Integrity
     i. Ensuring that users of data/resources are the persons they claim to be and that a message has not been altered

Encryption

```
USER A
Hello-> key -> 4H$D2
```

```
USER B
4H$D2 -> key -> Hello
```

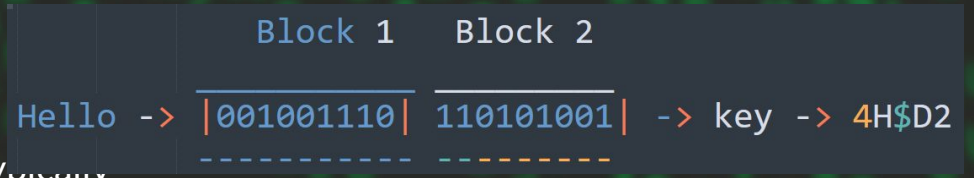


Cipher

- Cipher is a method for encrypting messages
- Encryption algorithms are standardized & published
- The key which is an input to the algorithm is secret
- **Key**: is a string of numbers or characters
- If same key is used for encryption & decryption the algorithm is called symmetric
- If different keys are used for encryption & decryption the algorithm is called asymmetric

Symmetric Algorithms

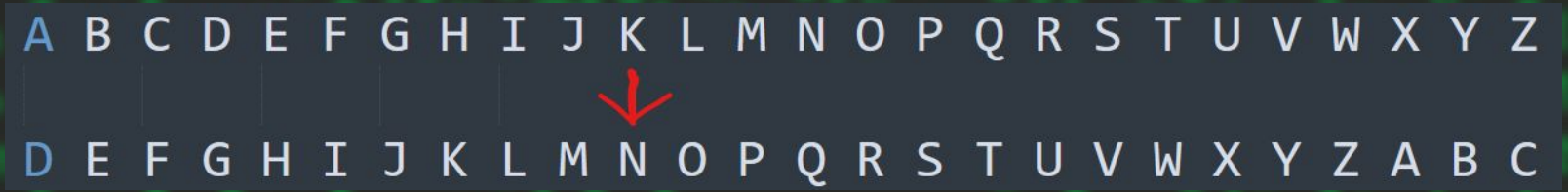
- Algorithms in which the key for encryption and decryption are the same are Symmetric
  - Example: Caesar Cipher
- Types:
  - Block Ciphers
    - Encrypt data one block at a time (typically 64 bits, or 128 bits)
    - Used for a single message
  - Stream Ciphers
    - Encrypt data one bit or one byte at a time
    - Used if data is a constant stream of information



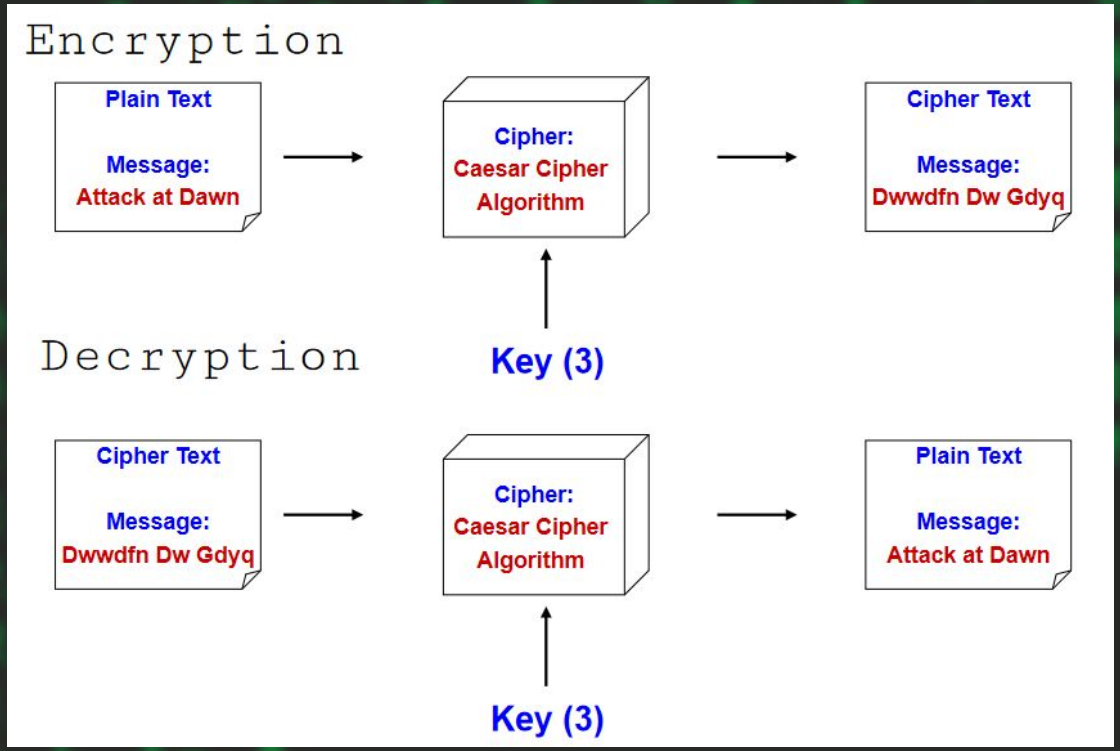
Substitution Ciphers

Caesar Cipher

Caesar Cipher is a method in which each letter in the alphabet is rotated by three letters as shown



...


Encryption
Plain Text
Message:
Attack at Dawn
Cipher:
Caesar Cipher
Algorithm
Cipher Text
Message:
Dwwdfn Dw Gdyq
Key (3)
Decryption
Cipher Text
Message:
Dwwdfn Dw Gdyq
Cipher:
Caesar Cipher
Algorithm
Plain Text
Message:
Attack at Dawn
Key (3)

Substitution Cipher
Using a key to shift alphabet

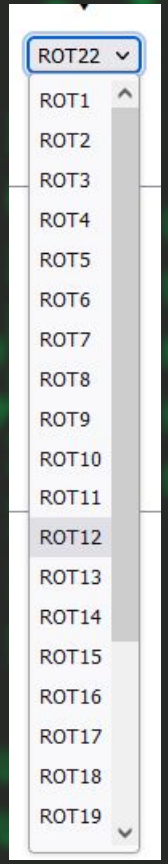
- Obtain a key to for the algorithm and then shift the alphabets
  - For instance if the key is ***word*** we will shift all the letters by four and remove the letters w, o, r, & d from the encryption
- We have to ensure that the mapping is one-to-one
  - no single letter in plain text can map to two different letters in cipher text
  - no single letter in cipher text can map to two different letters in plain text

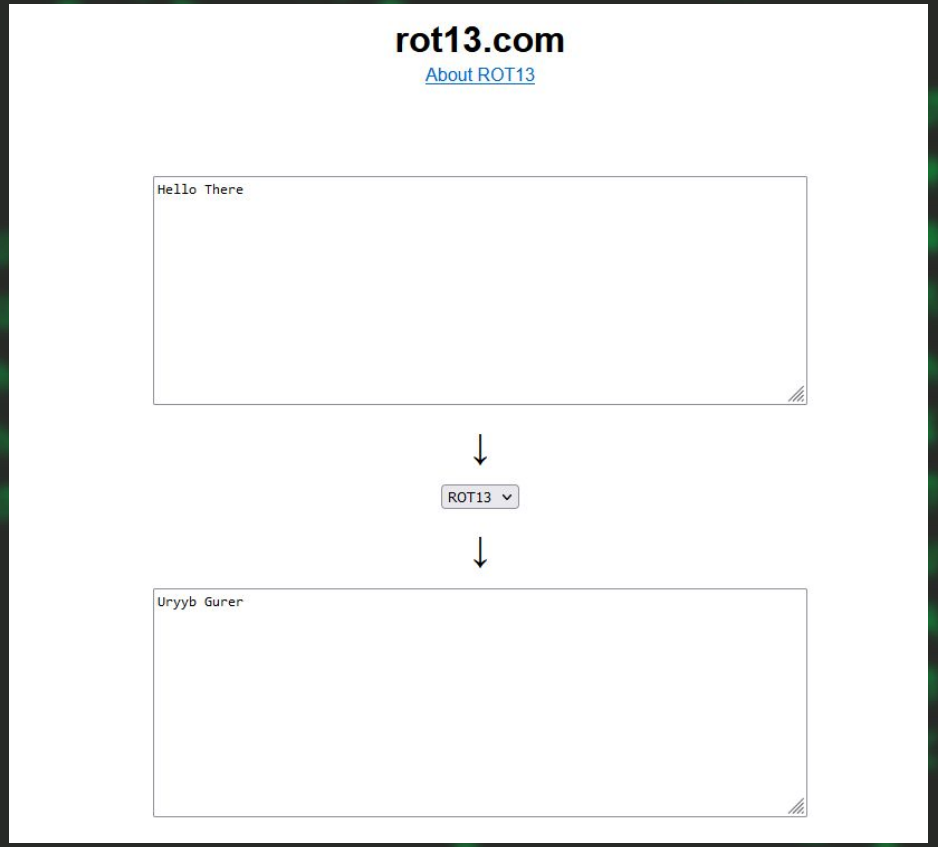
```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
                  |
H E Y A B C D F G I J K L M N O P Q R S T U V W X Z
```

```
message   =>   Cipher   =>   Encrypted
 BYE             HEY            EXB
```

Replacing the letters by 1 shift we can get different rotations. To do this we can use this website.

This encoding is called rot encoding





Website: rot13.com

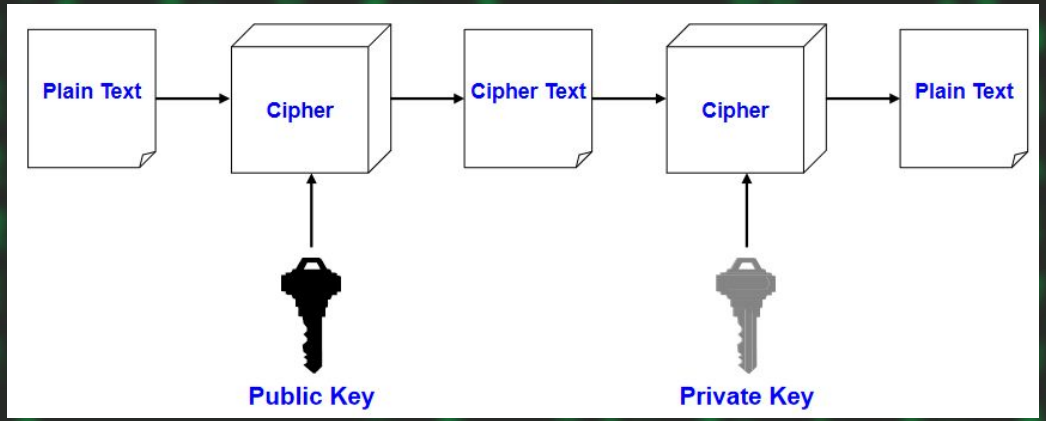
Limitation

- Any exposure to the secret key compromises confidentiality of ciphertext
- A key needs to be delivered to the recipient of the coded message for it to be deciphered
  - Some intruders can get the key and BOOM! No secret anymore.

Exercise 1

1. Change “Pass1233” text to caesar Cipher
2. What is the starting alphabet to produce “Rexler” text by shifting
3. What is the encoding method of “Rexler”
   a. Answer: Rot_
4. Change “Hello There” to cipher with a key to alphabet shift of “HACK”

Asymmetric Encryption



- Uses a pair of keys for encryption
  - Public key for encryption
  - Private key for decryption
- Messages encoded using public key can only be decoded by the private key
  - Secret transmission of key for decryption is not required
    - Public key can be exposed so, if i need to send you a message i just ask you for your public key and i will encrypt the message with your public key. When you get the ciphertext you can decrypt it with your private key.
  - Every entity can generate a key pair(private&public) and release its public key

Types of asymmetric enc.

- Two most popular algorithms are RSA & El Gamal
- RSA
  - Developed by Ron **R**ivest, Adi **S**hamir, Len **A**delman
  - Both public and private key are interchangable
  - Variable Key Size (512, 1024, or 2048 bits)
  - Most popular public key algorithm
  - It have a maths formulas for generating the keys.
- El Gamal
  - Developed by Taher ElGamal
  - Variable key size (512 or 1024 bits)
  - Less common than RS

Applications

One of example program that use asymmetric encryption is SSH.

When you Create/Config SSH on your computer it give u 2 keys, 1 public and 1 private, so when connection is established each hosts exchange their public key and store it in (known_hosts), then anytime they send data they will encrypt them with the public, and the host will decrypt it with its private key.

So if your Private Key got leaked 💀

```
❯ ls .ssh
id_rsa  id_rsa.pub  known_hosts
```

```
❯ cat .ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEA0dchR7K8GC4ssiuMKjBAYC8unOFb1jjX4/b8KqJADRCZg9mHmlLI
6oQLQsc7LN1o6QTfTWb6dIBa/5XVdANzT0U95+bUOLkgoW0hyw2OwZeeb5M1wAJyvNkRZg
c9tSYXUD+PUYbGpXkQ/O5vt+SLECA5XH3qPzSVXKQ+KPlWF14A5YAHMU6jm+uqPRrShkq2
4Fv1zTYsnhnBk+cJqKlOLUi6jwt0f0IGppUigAZVlKzii6gl4uOfktuk26iFsgaRXP2Qcg
CVL++nqP+j6OHeHjpVPJCOKShvA8VrpoW0N6HHM2TgUOT9m3ZtDMdNZjohQykIMTJVEKIv
HwJ0Rq42FV7/rKcsQ431951M5dk2iPjGhO3diA47wo2OBq4qGzpY8avrQuZzfiQ1KaNNiZ
DcdPprxyobnCuAyzylEbWMuqlntAR+IqB5t8LZUSVjS0vLhZ47rXuvlVqiRI3nw8VAiSe/
VnvxAgc8QcNVD7/4xcuxMYNog3Xk5YgxzNt4HBZpAAAFkFR+qsdUfqrHAAAAB3NzaC1yc2
EAAAGBANHXIUeyvBguLLIrjCowQGAvLpzhW9Y41+P2/CqiQA0QmYPZh5pSyOqEC0LHOyzd
aOkE301m+nSAWv+V1XQDc09FPefm1Di5IKFtIcsNjsGXnm+TNcACcrzZEWYHPbUmF1A/j1
GGxqV5EPzub7fkixAgOVx96j80lVykPij5VhdeAOWABzFOo5vrqj0a0oZKtuBb9c02LJ4Z
wZPnCaipTi1Iuo8LdH9CBqaVIoAGVZSs4ouoJeLjn5LbpNuohbIGkVz9kHIAlS/vp6j/o+
```

```
❯ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDR1yFHsrwYLiyyK4wqMEBgLy6c4VvWONfj9vw
A3NPRT3n5tQ4uSChbSHLDY7Bl55vkzXAAnK82RFmBz21JhdQP49RhsaleRD87m+35IsQIDlcfeo
GcGT5wmoqU4tSLqPC3R/QgamlSKABlWUrOKLqCXi45+S26TbqIWyBpFc/ZByAJUv76eo/6Po4d4
KQgxMlUQoi8fAnRGrjYVXv+spyxDjfX3nUzl2TaI+MaE7d2IDjvCjY4GriobOljxq+tC5nN+JDU
JWNLS8uFnjute6+VWqJEjefDxUCJJ79We/ECBzxBw1UPv/jFy7Exg2iDdeTliDHM23gcFmk= re
```

```
❯ cat .ssh/known_hosts
|1|asxFUVEIXM6KJjMj27PbttYEGM8=|YI+fefaWF7da8fwkPwwTvU2IZtw= ssh-ed25519 AAAAC3NzaC1lZD
gqiKT31/JhVUc6P7f+JQe
|1|PACAQFqZJMC2ncWPc7hoKJP6jlk=|y0clCEEqtCxiogJCReM8et0I+is= ecdsa-sha2-nistp256 AAAAE2
zdHAyNTYAAABBBIFtuu4PY8SUHVEcFf/QO9DEqRxaHlbm7l5aWtSMQTwhjFuRYpJRU/xsr2pQbtUJ0/yjo2zq62
|1|Wjlf5BIAYNgpbrZaa8A90/AZS5E=|lC47vDl8jnrq868PiT7Ay2xlPw0= ssh-ed25519 AAAAC3NzaC1lZD
osxpZ5Y333MqaJieK/1gF
```

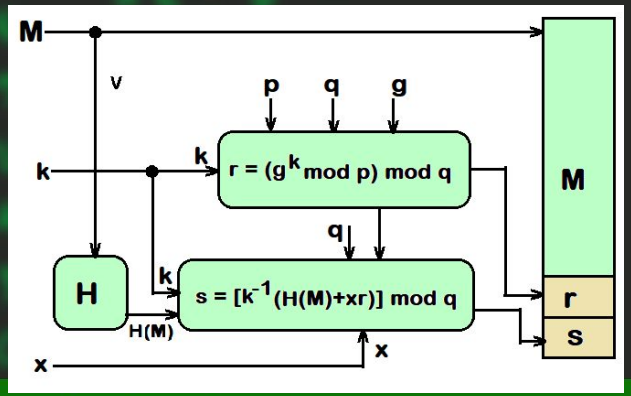
RSA, AES, DSA

Encryption:

Plaintext	$M < n$
Ciphertext	$C = M^e (mod\ n)$

Decryption:

Ciphertext	C
Plaintext	$M = C^d (mod\ n)$



$5 = 120 - 5*23 = (1*120) - (5*23)$

$3 = 23 - 4*5 = (1*23) - 4*[(1*120 - 5*23)] = (-4*120) - (21*23)$

$2 = 5 - 1*3 = [(1*120) - (5*23)] - 1*[(-4*120) + (21*23)] = (5*120) - (26*23)$

$1 = 3 - 1*2 = [(-4*120) + (21*23)] - 1*[(5*120) - (26*23)] = (-9*120) + (47*23)$

Figure 3. Calculation steps to obtain coefficient of 120 and 23 using decimal values

R1 = a – Q1*b

R2 = b – Q2*R1
= b – Q2*(a – Q1*b)
= - Q2 *a + (1 + Q1*Q2)*b

R3 = R1 – Q3*R2
= (a – Q1*b) – Q3*[- Q2 *a + (1 + Q1*Q2)*b]
= (1 + Q2*Q3)*a – (Q1 + Q3 + Q1*Q2*Q3)*b

R4 = R2 – Q4*R3
= [- Q2 *a + (1 + Q1*Q2)*b] – Q4*[(1 + Q2*Q3)*a – (Q1 + Q3 + Q1*Q2*Q3)*b]
= - (Q2 + Q4 + Q2*Q3*Q4)*a + (1 + Q1*Q2 + Q1*Q4 + Q3*Q4 + Q1*Q2*Q3*Q4)*b

R5 = R3 – Q5*R4
= [(1 + Q2*Q3)*a – (Q1 + Q3 + Q1*Q2*Q3)*b] – Q5*[- (Q2 + Q4 + Q2*Q3*Q4)*a + (1 + Q1*Q2 + Q1*Q4 + Q3*Q4 + Q1*Q2*Q3*Q4)*b]
= (1 + Q2*Q3 + Q2*Q5 + Q4*Q5 + Q2*Q3*Q4*Q5)*a – (Q1 + Q3 + Q1*Q2*Q3 + Q5 + Q1*Q2*Q5 + Q1*Q4*Q5 + Q3*Q4*Q5 + Q1*Q2*Q3*Q4*Q5)*b

R6 = R4 – Q6*R5
= [- (Q2 + Q4 + Q2*Q3*Q4)*a + (1 + Q1*Q2 + Q1*Q4 + Q3*Q4 + Q1*Q2*Q3*Q4)*b] – Q6*[(1 + Q2*Q3 + Q2*Q5 + Q4*Q5 + Q2*Q3*Q4*Q5)*a – (Q1 + Q3 + Q1*Q2*Q3 + Q5 + Q1*Q2*Q5 + Q1*Q4*Q5 + Q3*Q4*Q5 + Q1*Q2*Q3*Q4*Q5)*b]
= (Q2 + Q4 + Q2*Q3*Q4 +Q6 + Q2*Q3*Q6 + Q2*Q5*Q6 + Q4*Q5*Q6 + Q2*Q3*Q4*Q5*Q6)*a – (1 + Q1*Q2 + Q1*Q4 + Q3*Q4 + Q1*Q2*Q3*Q4 + Q1*Q6 + Q3*Q6 + Q1*Q2*Q3*Q6 + Q5*Q6 + Q1*Q2*Q5*Q6 + Q1*Q4*Q5*Q6 + Q3*Q4*Q5*Q6 + Q1*Q2*Q3*Q4*Q5*Q6)*b

Figure 4. Formulas in obtaining the inverse values

3 Terms of Cryptography

1) Encoding/ decoding
   a) This is a method of creating Cipher text with out using any key
   b) This can be done by doing math on the given input/substitution
      i) Examples: base64,base32,rot…
2) Encrypting/Decrypting
   a) This is method of creating Cipher text with keys.
   b) To decrypts this kind u need to have the private key
      i) Example: DES,AES,RSA
3) Hashing
   a) This is a method of creating Cipher text with respect to a created hash
   b) To reverse the hash, you just search for some match, you don't decrypt/decode it.
   c) **Salt**: is a random string used for data modification for password protection, This can be adding some text as prefix/suffix
      i) Example: MD5,sha254,...

Kinds of encodings/encryptions

- Base2 01100010 01110010 01100101 01100001 01101011 01101001 01110100
- Base8 142 162 145 141 153 151 164
- Base16 62 72 65 61 6b 69 74
- Base32 MJZGKYLLNF2A====
- Base58 4jP4KDubX1
- Base62 22udqyscMu
- Base64 YnJlYWtpdA==
- Base85 @WH$gCM@k
- Base91 %zmfv;:YH
- URL encode: hello%20there%20%3F
- Md5: 5d41402abc4b2a76b9719d911017c592
- Sha1: aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
- Rot : Uryyb, Frphevgl Grfgref => look for some random word that looks rotated

tools

- There are lots of encodings/encryption
- To identify this we will need some tools/sites
- Tools:
  - hashid
    - hashid <hash>
  - Cyber chef (web)
  - Tunnelsup (web)

```
┌──(nathan㉿Nathan)-[~]
└─$ hashid 5d41402abc4b2a76b9719d911017c592
Analyzing '5d41402abc4b2a76b9719d911017c592'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
```

Tunnels Up

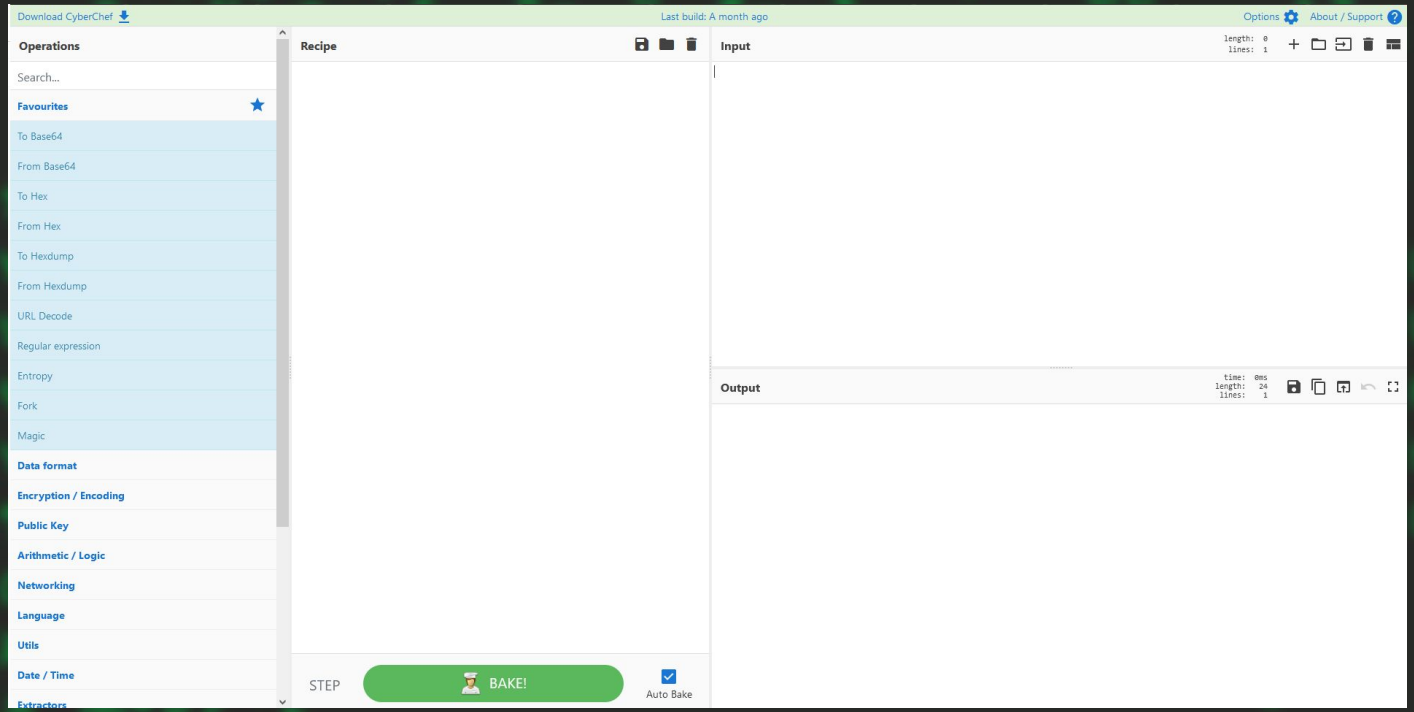
- THis will help You analyze and determine what that hash is based on the bit length and the base.

tunnelsup.com/hash-analyzer/

CyberChef

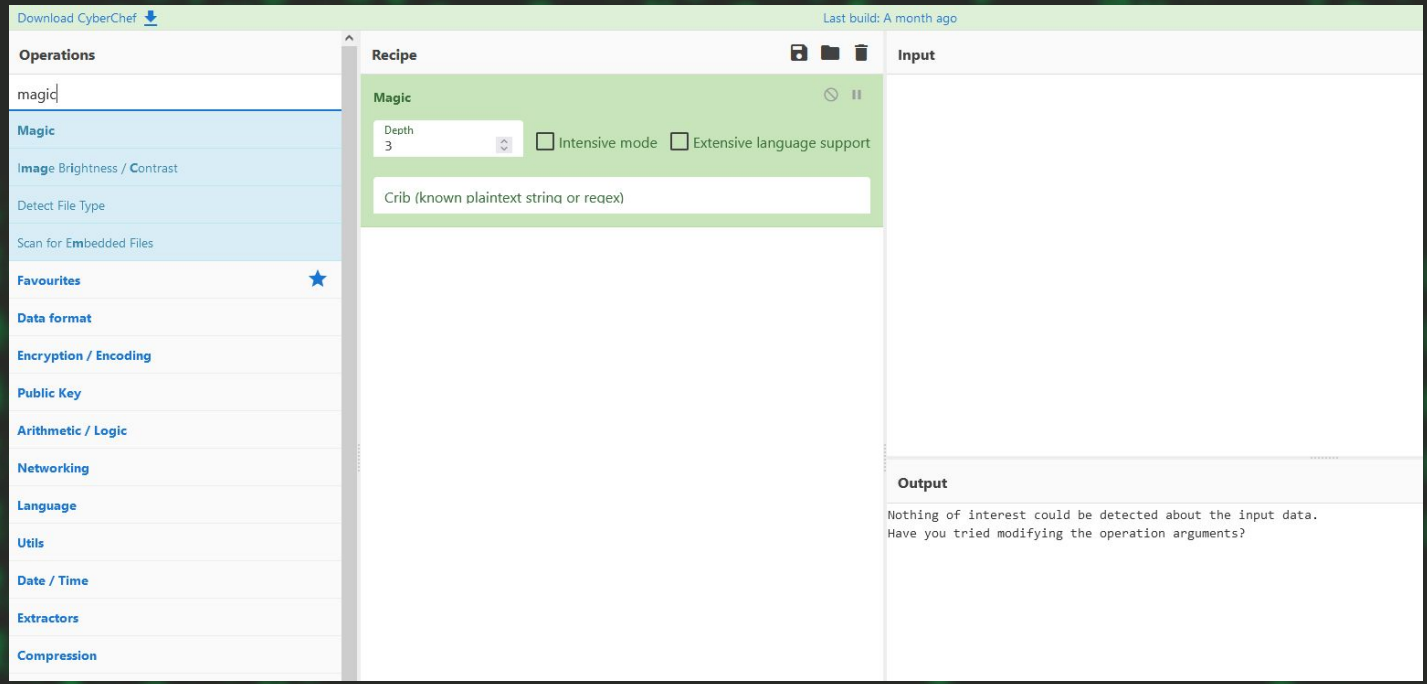
- Goto google and type cyberchef
- Click on the 1st link.

- Link : https://gchq.github.io/CyberChef/



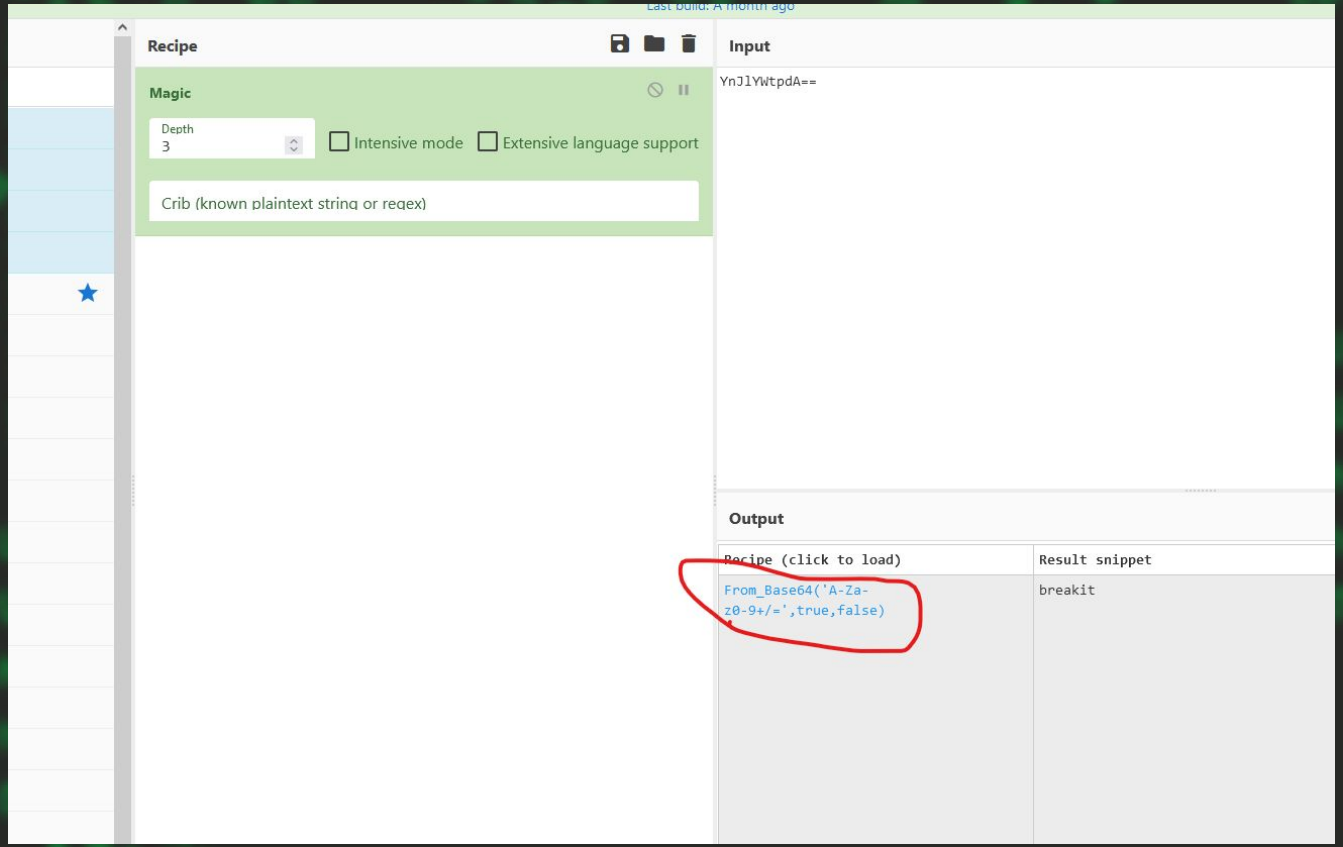
...

- Search for magic
- Drag and drop it, to the recipe.



...

- Add your text to the input
- Look at the output it is the guess of what the hash can be

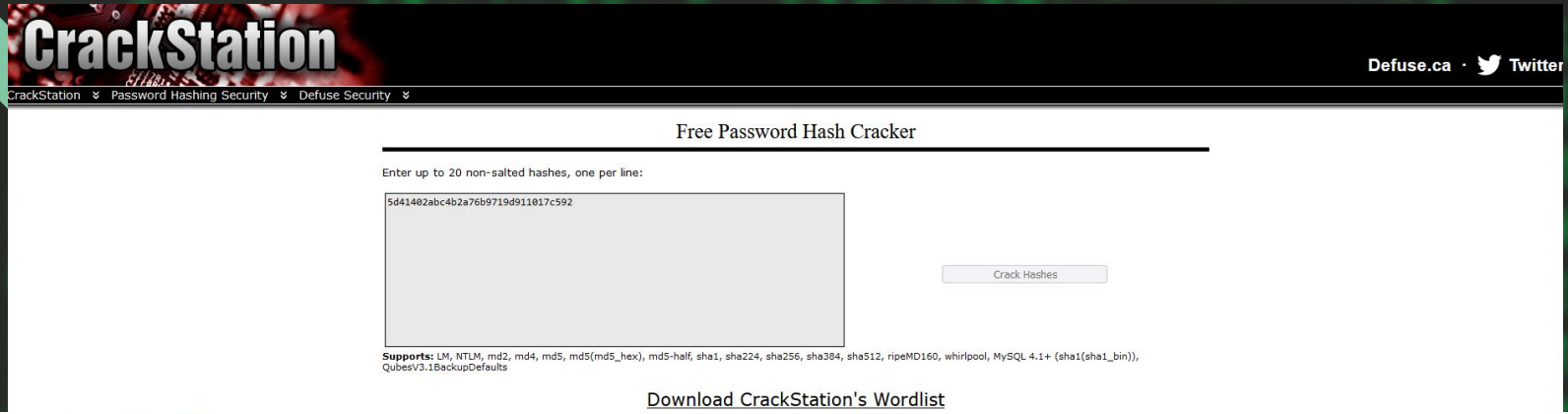


decoding/decrypting

There are so, many way to reverse some hashes/ciphers.

- Hashes
  - Craskstation.net(non-salted)
  - Own cracking(google the name)
- Encodings
  - CyberChef

Crackstation



How CrackStation Works

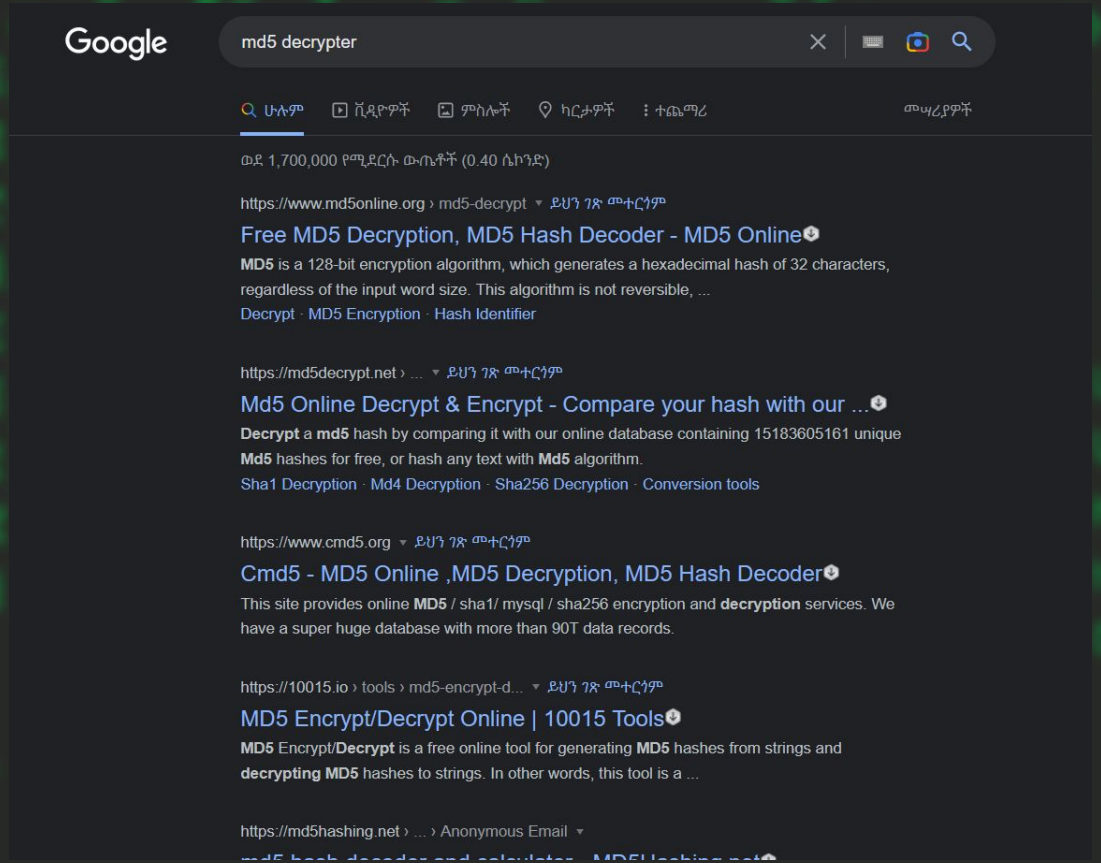
CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our hashing security page.

Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries here, and the lookup table implementation (PHP and C) is available here.

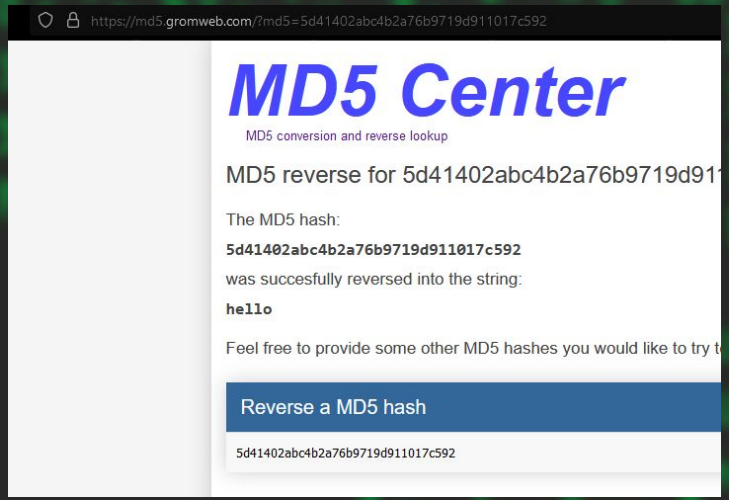
Own pages

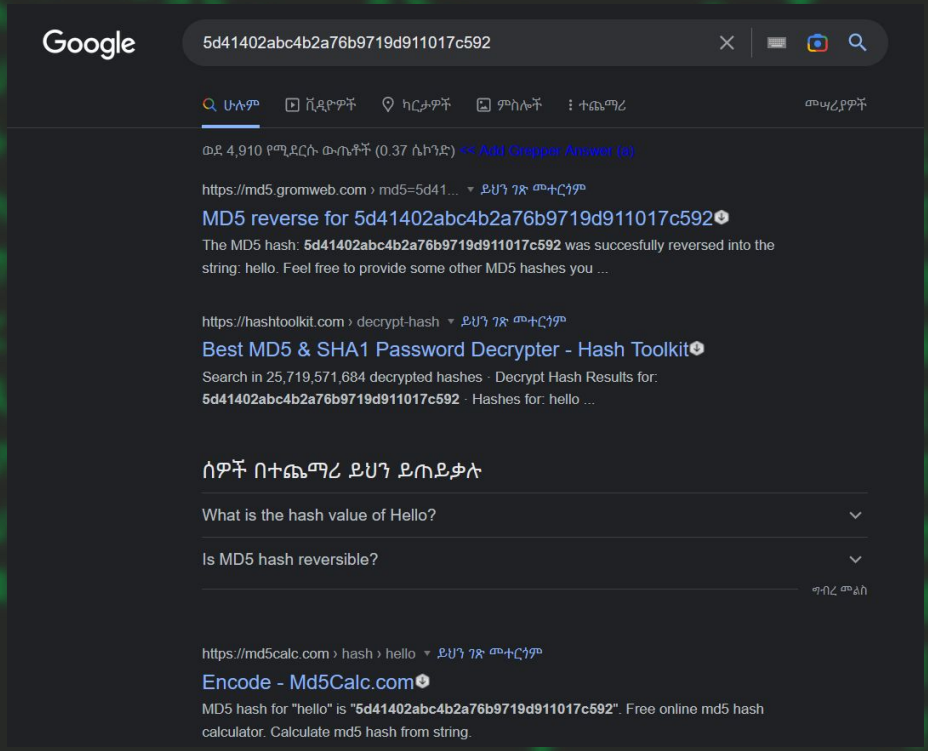
By searching the hash type you can get the decoder/decrypter.



also ...

Searching the hash is Good

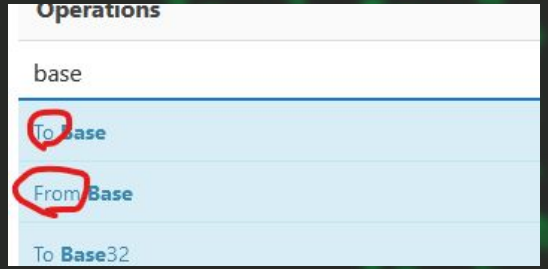


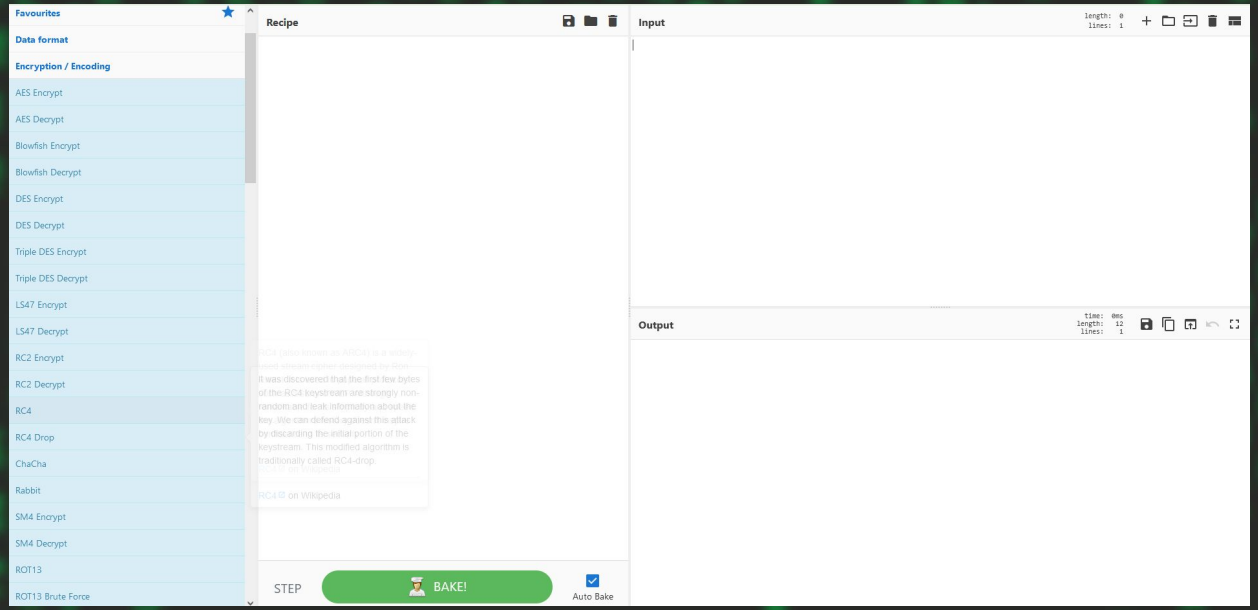


Cyber chef

By searching any encryption you can decode/decrypt it.

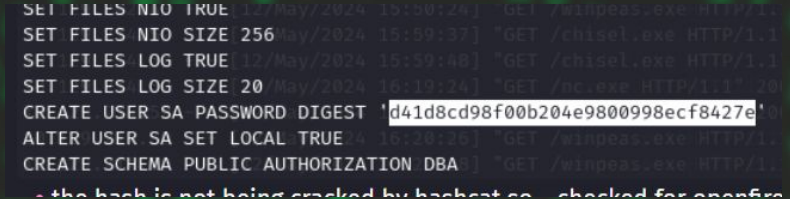
- We use from to decrypt
- We use to to encrypt

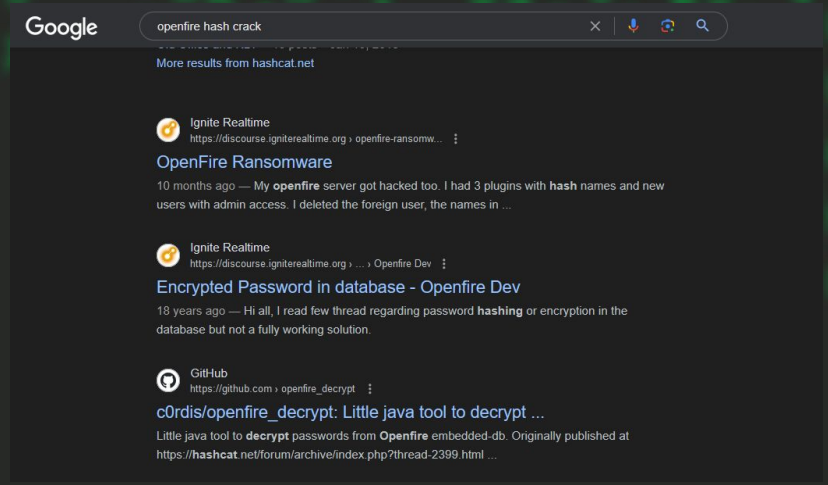




Identifying Unknown Hashes

- Sometimes when you do penetration tests you will get some Hash. these hashes are not normal hashes they are generated by some Platform/Software. So to Crack this hash.
  - Identify the Software which the hash Generated with.
    - Ex: The Hash can be generated by some Software called Openfire
  - Then Try to Search Some Cracking Scripts made for this hash





Wordlists

- Wordlists are a normal text file that contains Different Words that can be used to match hashes, or for checking some parameters repeatedly using some loops.

```
      26 May 13 03:10 amass -> /usr/share/ama
      25 May 13 03:10 dirb -> /usr/share/dirb
      30 May 13 03:10 dirbuster -> /usr/share
      35 May 13 03:10 dnsmap.txt -> /usr/shar
      41 May 13 03:10 fasttrack.txt -> /usr/s
      45 May 13 03:10 fern-wifi -> /usr/share
      28 May 13 03:10 john.lst -> /usr/share/
      27 May 13 03:10 legion -> /usr/share/le
      46 May 13 03:10 metasploit -> /usr/shar
      41 May 13 03:10 nmap.lst -> /usr/share/
      13 May 17 13:09 rexder.virus
139921507 May 13 03:10 rockyou.txt
      19 May 13 03:03 seclists -> /usr/share/
      39 May 13 03:10 sqlmap.txt -> /usr/shar
    4096 May 18 05:40 thinclient_drives/
      25 May 13 03:10 wfuzz -> /usr/share/wfu
      37 May 13 03:10 wifite.txt -> /usr/shar
```

```
rexder@station ~> head -n 30 rockyou.txt
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
babygirl
monkey
lovely
jessica
654321
michael
ashley
qwerty
111111
iloveu
000000
```

```
rexder@station ~> head -n 30 seclists/Discovery/Web-Content/common.txt
.bash_history
.bashrc
.cache
.config
.cvs
.cvsignore
.forward
.git
.git-rewrite
.git/HEAD
.git/config
.git/index
.git/logs/
.git_release
.gitattributes
.gitconfig
.gitignore
.gitk
.gitkeep
.gitmodules
.gitreview
.history
```

Custom Wordlists

- We can Create our own Wordlists, We can Create text file and add our highly usable words or we can use tools like
  - Cewl
  - Cupp

```
┌──(root㉿kali)-[~]
└─# cewl http://vulnweb.com -d 3
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Acunetix
learn
more
the
http
vulnweb
com
Review
scanner
topic
SQL
site
for
PHP
you
Web
Vulnerability
Scanner
websites
test
Apache
MySQL
IIS
ASP
Microsoft
Server
applications
```

```
root@kali:~/InstalledItem/cupp# python3 cupp.py -i
                          ___________
  cupp.py!                # Common
      \                   # User
       \   ,__,          # Passwords
        \  (oo)____       # Profiler
           (__)    )\
              ||--||      [ Muris Kurgas | j0rgan@remote-exploit.org ]
                          [ Mebus | https://github.com/Mebus/]

[+] Insert the information about the victim to make a dictionary
[+] If you don't know all the info, just hit enter when asked! ;)

> First Name:

[-] You must enter a name at least!
> Name:

[-] You must enter a name at least!
> Name: a
```

Exercise 2

1) What is the hashing algorithm of this text “aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d”
2) What is the decoded text of “SGVsbG8sIFNlY3VyaXR5IFRlc3RlcnM=”
3) What is the decoded text of “KUZFM2TELBFHAZCINNTVMR2WPJSEOVTZMN3T2PI=”
4) What is the decrypted text of “21232f297a57a5a743894a0e4a801fc3”

Python for cryptography

- We can use programming to do tools that can do our own encryption and encoding hash type
- There are so many methods, even you can do the encoding/decodeing for the base64…
- You just need to understand the maths.
- Now i will show u simple XOR'ing example
  - What is XOR?

Pseudo code

```
1 encode function
2 string in number(ord) ^ key
3 result to hex(hex)
4 stored to variabel called encrypt_hex
5 Displayed
```

```
1 decode function
2 hex to unicode and stored on variabel called hex2uni
3 hex2uni in number(ord) ^ key
4 change the result to alphabetic character
5 stored to variabel called decrypt_text
6 Displayed
```

```python
# Encrypting code
def encrypt():
    msg=input("message: ")
    key=input("key: ")
    encrypt_hex = ""
    key_itr=0
    for i in range(len(msg)):
        # Change the msg into unicode decimal and XOR it with the key
        deciTEXT = ord(msg[i]) ^ ord(key[key_itr]) # iterating msg and key
        key_itr+=1
        # check if the key_itr is greater than length of key
        if key_itr >= len(key):
            # Once all the key's letter used repeat the key
            key_itr=0
        # return the decimal to hexadecimal value
        # and remove the 1st 2 digits(0x34 -> 34)
        encrypt_hex += hex(deciTEXT)[2:]

    print(f"Message Encrypted Successfully!\nmessage: {encrypt_hex}")
```

```python
# Decrypting code
def decrypt():
    msg=input("message: ")
    key=input("key: ")
    hex2uni = ""
    key_itr=0
    decrypt_text=""
    # Fetching the hex data to unicode
    for i in range(0,len(msg),2):
        # bytes.fromhex() changes the hex to unicode
        # it adds b'' to the value so to remove that .decode('utf-8')
        hex2uni+=bytes.fromhex(msg[i:i+2]).decode('utf-8')

    for i in range(len(hex2uni)):
        # XORing the hex2uni and key to get the original data
        # a^b =c    |    c^b=a
        # msg^key=hex2uni    |    hex2uni^key=msg
        temp = ord(hex2uni[i]) ^ ord(key[key_itr])
        # Converting the unicode text to characters
        decrypt_text+=chr(temp)
        # check if the key_itr is greater than length of key
        key_itr+=1
        if key_itr >= len(key):
            key_itr=0
    # Display the text
    print(f"The Decrypted Message is:\n{decrypt_text}")
```

```
50  print("Welcome to rexEncrpter.")
51  print("==========================")
52  # Accepting input
53  user=input("What do you want to do \n 1) Encrypt \n 2) Decrypt\n>>")
54
55  # Validating the input
56  if user == '1':
57      encrypt()
58  elif user == '2':
59      decrypt()
60  else:
61      print("ERROR, No input!")
```

output

```
Welcome to rexEncrpter.
========================
What do you want to do
 1) Encrypt
 2) Decrypt
>>1
message: This is Top Secret, GTST is Started last time!
key: 0102
Message Encrypted Successfully!
message: 64595941105843126 45e4012635453405545 1c12765636610584312634551404454541 25c504
3461045595f5510
```

```
Welcome to rexEncrpter.
========================
What do you want to do
 1) Encrypt
 2) Decrypt
>>2
message: 64595941105843126 45e4012635453405545 1c12765636610584312634551404454541 25c504
3461045595f5510
key: 0102
The Decrypted Message is:
This is Top Secret, GTST is Started last time!
```

Base64 decode/encode

```python
import base64
msg=input("text: ")
encoded=base64.b64encode(bytes(msg,'utf-8'))
print(encoded)

decoded=base64.b64decode(encoded)
print(decoded)
```

```
text: nathan
b'F0aGF'
b'nathan'
```

Obfuscation

- In software development, *obfuscation* is the act of creating source or machine code that is difficult for humans or computers to understand.
- As we know High level programming lang. are easy to understand, so if hackers got your code he can read it, but to make it more difficult we use this technique
- This was the code obfuscated

```
function hi() {
  console.log("Hello World!");
}
hi();
```

```
(function(_0x59f190,_0x497c58){var _0x263e02=_0x2d87,_0x13b460=
(-parseInt(_0x263e02(0x144))/0x2)+parseInt(_0x263e02(0x139))/0x3*(
/0x5*(-parseInt(_0x263e02(0x143))/0x6)+parseInt(_0x263e02(0x13c))/
/0x9*(-parseInt(_0x263e02(0x145))/0xa)+-parseInt(_0x263e02(0x141))/
else _0x13b460['push'](_0x13b460['shift']());}catch(_0x4c235c){_0x13b
_0x2d87(_0x488b23,_0x1af22c){var _0x28ae4c=_0x28ae();return _0x2
_0x3597ee=_0x28ae4c[_0x2d87fe];return _0x3597ee;},_0x2d87(_0x48
console[_0x530925(0x13a)](_0x530925(0x13e));}hi();function _0x28ae(
['log','16KbcLcL','875896XaJYgl','29050nJjyOa','Hello\x20World!','2515
```

...

My python code

```python
# Encrypting code
def encrypt():
    msg=input("message: ")
    key=input("key: ")
    encrypt_hex = ""
    key_itr=0
    for i in range(len(msg)):
        # Change the msg into unicode decimal and XOR it with the key
        deciTEXT = ord(msg[i]) ^ ord(key[key_itr]) # iterating msg and key
        key_itr+=1
        # check if the key_itr is greater than length of key
        if key_itr >= len(key):
            # Once all the key's letter used repeat the key
            key_itr=0
        # return the decimal to hexadecimal value
        # and remove the 1st 2 digits(0x34 -> 34)
        encrypt_hex += hex(deciTEXT)[2:]

    print(f"Message Encrypted Successfully!\nmessage: {encrypt_hex}")


# Decrypting code
def decrypt():
    msg=input("message: ")
```

obfuscated

```
#pip install pycryptodome
import random ,base64,codecs,zlib;pyobfuscate=""

obfuscate={'(https://pyobfuscate.com)*(public_key)':""E)91<17`;>!I&#9InNi$;8U?TgzC(ryKdYI941K~BJ+cDlcLUyxtfB}1cd-KpsL8]K&
E0dHrsB)F)g^Swe`WR5aTWTKpsD<Z-i2@hh*5cK46{Jk@VpTrG7?o(LPfBAPi?DMhK?n3E|@4Lh5OZ%j{YN8&
f>F5<@_@d5Bb$+>pO`+$`(VAWcdfd`)>,JFG2P9II2B|ie#dwIV-2Rgb_uod0XtaS4Bk@HH9oo3kWPN`uDIYng5fpSi_^=S||i36Z$+ia9J_SKYAE1sQa1ID&
NU*PQ@^9J3_GN$i2fb>KYCyi2nIcoi-U6cgqn<QX5{Ps#weQBJm*2q@b8Ko^JDrMPBBq~eB*sWGy{0Vo^dxX6%)g`z!
<oHkrWAcvgIzKWbRl6Mhw1nEj#9TmMIvm)nDky4hYI3lw5ot9%<ZGXeDZ_<qw9**D2kH0qth5a5QQY86J4LKgTVyW%vAyZO~xROwHlPPMjpg(R*Y7cRqW2x95Q-
s7IGuAXSn79L@UKeyJ0t(&mhH~G6pZ{<C-ULkldqPIr{dSs6%aV|2s(ZC@IGEk|7<suR6zIcLaP2Qr)lgZ#%}4Qyaik=`9YEO&Sg_?M?Jlw(%oWRK&NMfmQG_|>C&
jaNKrqGse+(4Ar$4BB&Bnjo;tb9m7*WhfflFn)A4RYNJvP3e}836C(<q>Z)2c0t^);{pe}&ECsc4`k8nO0t?20&elI7kaPg&
7iK1h4Q$c1~#6>XfadS)Lr>N~#6Zhb}h(pMK=5U|bVI5>a7E2P;,YDiAv*G8GLweY*4&4OKrPAf&I8G6`4XA4ObG#;$5RI4#o%m1D1gHq+3ql=`4;W<Y+(wdyu>?NqR^;
TP2vVEYb?XP_U`;{5Gh&}=D(Q;+1@grQTUk~>vs#I-K%1q`+Q|~F<JCeblx>PbgBfN+nEoeeBmZ?ES4>>?40c&8T$&ZFQh7%1M-pJtv<B*,
DUZbAzh(d8Cr7Iz90X)to#a{#YFf1Z04hXX0QAMvU}m_lgxN7%5;xQoax0*S3pUj_?-pQTvX(33b*uC$P<8g5lmI)(oWey;>^#>K9G6_B*h;
E6$}H$a9cRk`Rh}NPAs5n<okId7G1GtD6=#-Jg9M$WonJLj$PQ6q*c0~m$ePJ#4YK@8tMghke&
C<DNdJRZ2TlGgvEO]raBUCnk=YvFMzvNo_3lab>%mhx05u?PaXf=bv*P}-bOFz$C%zC5{{DS8wpUISJGL^C8P5`0e^SPC%?jA<|3f}E40sug<Faq-
NEA1H3l=e7iruDS_G{q)6Rgqn9VemRmks$|7n1T*PZq#TSYB9lRiYXYFZX8D(7MVG%OsS`mEB`RLVpJwowZ$XW3RrT+^#RgKoo&
rXt9Ukd2crA>~ktWOfVSBL1?`}*N8#7FSGc0oO|k>GBDbCZ;8UGcFgg$*d+uN6aDyBI?8UALWe4vM%NGRfDlkFKs`{F3xcw^a#%l)0Hh=2d4{Htysz$K=kvZuX$&
LH8HjGy^E=0$h&#D4_us1v-<NzM#}W3MZEQ^<=xi&L9LPz1Uq^r?wjNk$+UZ{?MACM*Nlvs6xV>zqO;
h%uodVSBNagg+o|Kmdr#4QVSTk8CI_NHqa}}5=$$B5>QqTla~DqZpOzr2CJh`zXXlrDZCCo0NZe^~RFi^Pm$~`(=PkuEh5HAKBI$N4UD-CQvnSUX4I@^=Jgv_;
bXZZbn3vMbY$jRjEgRYSA?*%bnoFIe7C2abR`HvL@T%)?bSJ=_WKsXV5++-bEyamHa$--z#W3D67A+%lv6`4;9t9V(ZRKQk^2$@jqSI_&os&)1ig`+l(bj6?$%D<Qj;
AzT{^pyyvr8ElGjLVlct#'l3Feqi-~tzF&K(A;B7(t-ohpZE%4SZGW{yYtR{z0kONuE)+3@Wy_`LO>)%{x<EQh>^jBIT8c)l)EBDe-
4q{Fv`Tp`Ih}6wS%#csSuR0FNODN(BBXcfj`7No(-E>L81F@EmR3*6u-R#$A{U?3z94S4NLhA-_`bYzNu(OLv1w)VliVa~T&$bTfrs7k;Pq=G2B%aSa&
5u@ss_PO<N3OYW*>%2;}DP0@@wN7xYeF&Vk8Eftailm8J;691*mox)OGbjuVHw<Ikr3Kh+O#jvmQk3;
mSl(08j=I+JiRpm$Y%fCA{qb^uuZ#F<{j%g7YGqYZ5OwJAUKH^uW-Si)YX|LKmN_C>xj`MKeU=Ohd9=QhBDg$K_3y#D4D)>2GSraiAqcg;lA<Z{9YACclEScxF>6&
2#5baFz7w?Ze1dXF@woe&cLtgr$(xG~fK(e^Xjp^N2*$j@%3y7S%OH|l6_=4|gmZAy1vHK^23AdtD-
5OZG6hi6caYTkLN`VD`8%o0TQ946fDh+Ns^yw?E|E3`jv2LOp?cb(O>ykujHVLXVlYKj#V^-ukGiusaH)OzE8X5{90blvMZ-
```

Website:
https://pyobfuscate.com/pyd

Exercise 3

1. Write a program that accepts a phone number, then encode it by multiplying the phone number by 123456. Also do the decoder
   a. Accepts only 251… numbers dot accept 09…
2. Do the caesar cipher algorithm with python
   a. Accept 1 numbers(string msg)
   b. 1 variable for the storing the changed value
   c. Iterate with the message accepted
      i. For every string change it to unicode decimal and add 3 to it
      ii. Then change the unicode decimal to character
      iii. Add the character to the storing variable
   d. Display the changed variable
      i. “The encoded Text is: … “

CLASS IS OVER

1. Ask question
2. DO the note(read more)
3. PRACTICE PRACTICE PRACTICE