



# Cryptography

S2Day5crypto.md



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

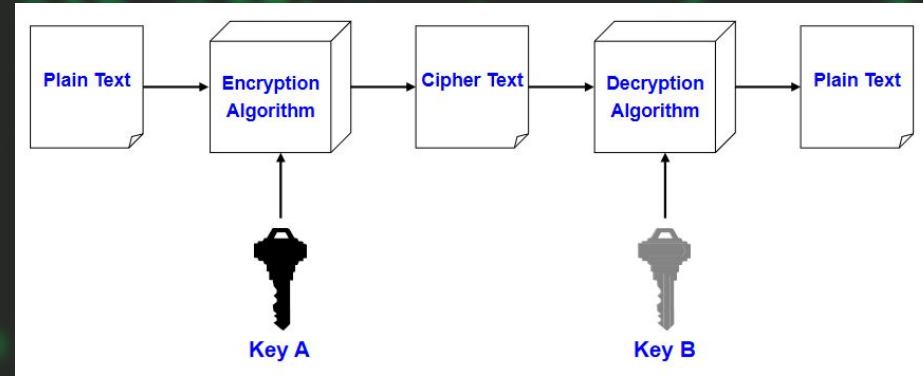
# Encryption

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

	Block 1	Block 2	
Hello ->	001001110	110101001	-> key -> 4H\$D2
	-----	-----	

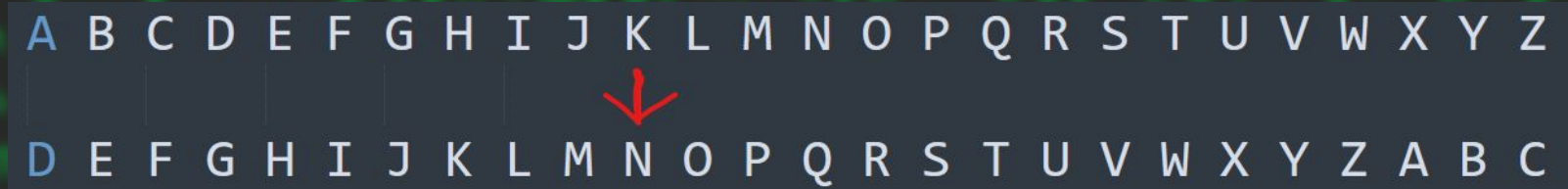
Hello -> |0|0|1|0|0|1|1|1|0|1|1|0|1|0|1|0|0|1| -> key -> 4H\$D2



# Substitution Ciphers

## Caesar Cipher

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



The diagram illustrates the Caesar Cipher rotation. It consists of two rows of letters. The top row is the standard alphabet: 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. The bottom row shows the alphabet shifted three positions to the right: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C. A red arrow points from the letter 'K' in the top row down to the letter 'N' in the bottom row, indicating a shift of three positions.

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

...

## Encryption



## Decryption



Key (3)

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

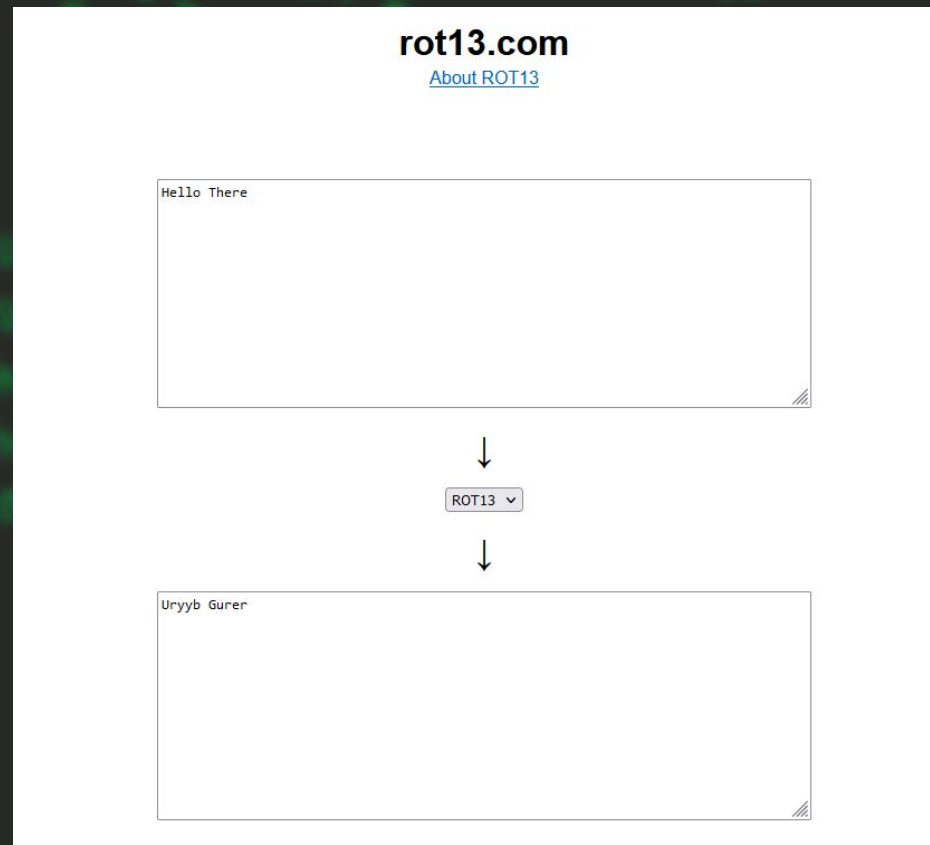
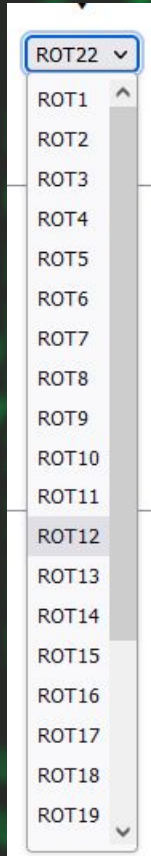
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	E	Y	A	B	C	D	F	G	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z

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](https://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.





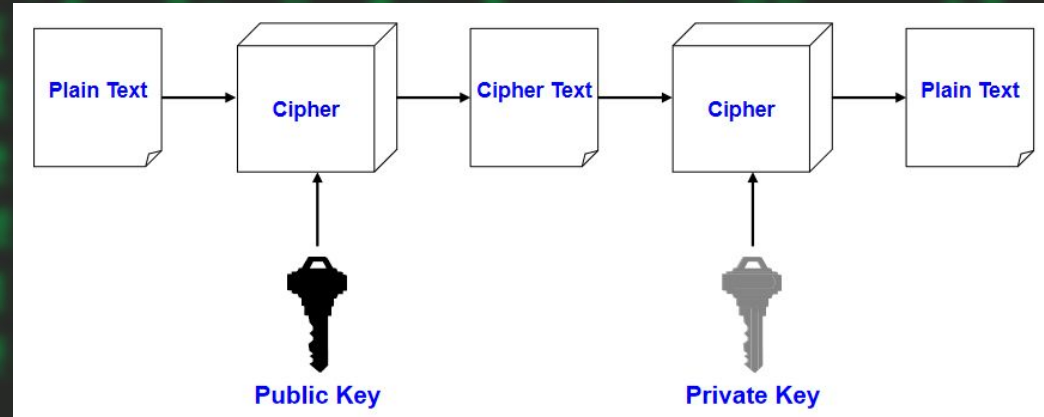
# Exercise 1

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 Rivest, Adi Shamir, Len Adelman
  - Both public and private key are interchangeable
  - 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



# 3 Terms of Cryptography

## 1) Encoding/ decoding

- a) This is a method of creating Cipher text with our 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.
  - 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 YnJIYWtpdA==
- 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

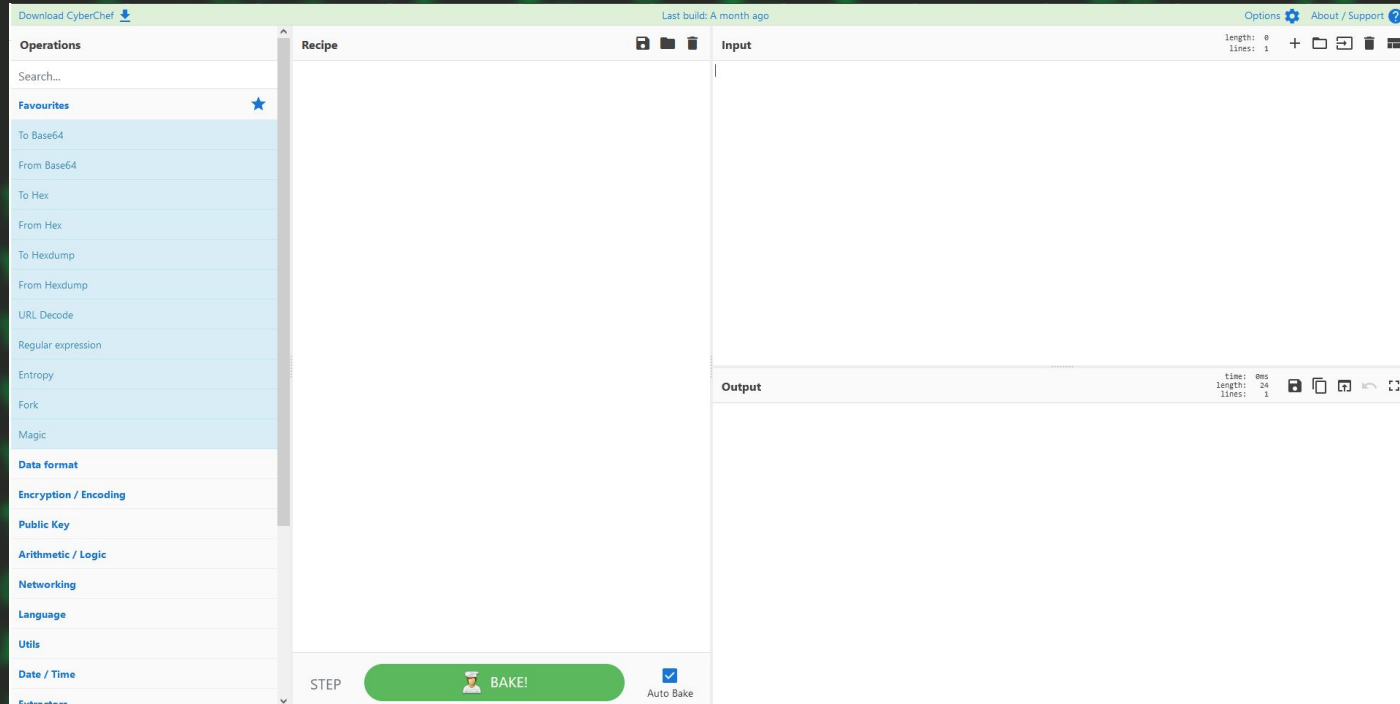
- There are lots of encodings/encryption
- To identify this we will need some tools/sites
- Tools:
  - hashid
    - hashid <hash>
  - Cyber chef (web)

```
(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
```

...

- 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.

The screenshot shows the CyberChef web application interface. On the left, a sidebar lists various operations. The 'Magic' operation is selected and highlighted in blue. Below it are other operations like 'Image Brightness / Contrast', 'Detect File Type', and 'Scan for Embedded Files'. The main area is titled 'Recipe' and shows the configuration for the 'Magic' operation. It includes a 'Depth' dropdown set to '3', and two checkboxes: 'Intensive mode' and 'Extensive language support', both of which are unchecked. Below these is a text input field labeled 'Crib (known plaintext string or regex)'. On the right side of the interface, there is an 'Input' section and an 'Output' section. The 'Output' section contains the text: 'Nothing of interest could be detected about the input data. Have you tried modifying the operation arguments?'. At the top right of the interface, it says 'Last build: A month ago'.

...

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

The screenshot shows the 'Recipe' application interface. The 'Input' section contains the text 'YnJlYWtpdA=='. The 'Output' section is a table with two columns: 'Recipe (click to load)' and 'Result snippet'. The first row in the table shows the recipe 'From\_Base64('A-Za-z0-9+/'',true,false)' circled in red, and the result snippet 'breakit'.

Recipe (click to load)	Result snippet
From_Base64('A-Za-z0-9+/'',true,false)	breakit





# 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

## CrackStation

CrackStation Password Hashing Security Defuse Security

Defuse.ca · Twitter

### Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

5d41402abc4b2a76b9719d911017c592

Crack Hashes

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1(sha1\_bin)), QubesV3.1BackupDefaults

### Download CrackStation's Wordlist

#### 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.

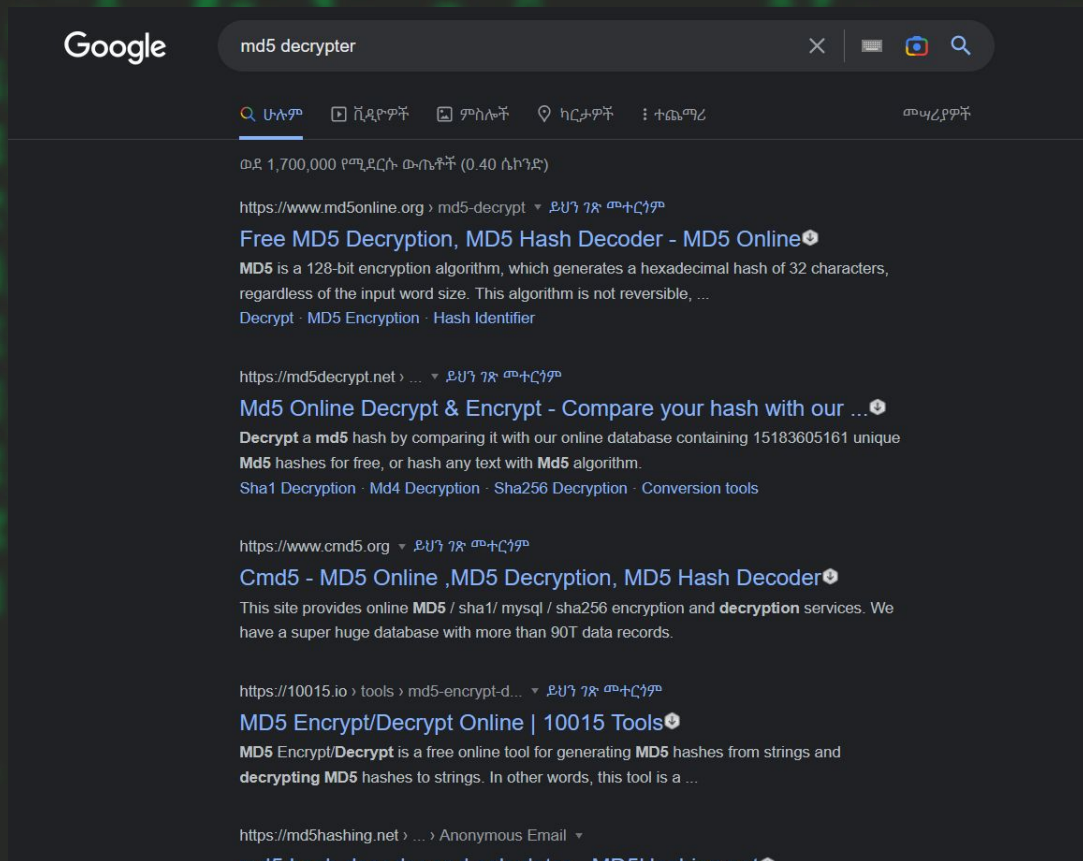
You can download CrackStation's dictionaries [here](#), and the lookup table implementation (PHP and C) is available [here](#).

Last Modified: May 27, 2019, 8:19am UTC  
Page Hits: 45757251  
Unique Hits: 8958969

crackstation.net | Defuse Security | Zcash | Secure Pastebin | Source Code

# Own pages

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



also ...

Searching the hash is Good

https://md5.gromweb.com/?md5=5d41402abc4b2a76b9719d911017c592

## MD5 Center

MD5 conversion and reverse lookup

MD5 reverse for 5d41402abc4b2a76b9719d911017c592

The MD5 hash:  
**5d41402abc4b2a76b9719d911017c592**  
was successfully reversed into the string:  
**hello**

Feel free to provide some other MD5 hashes you would like to try to

Reverse a MD5 hash

5d41402abc4b2a76b9719d911017c592

Google

5d41402abc4b2a76b9719d911017c592

ሁሉም ስራዎች ካርታዎች ምስሎች ተጨማሪ መሣሪያዎች

ወደ 4,910 የሚደርሱ ውጤቶች (0.37 ሰከንድ) << Add Gnipper Answer (a)

https://md5.gromweb.com › md5=5d41... › ይህን ገጽ መተርጎም

### MD5 reverse for 5d41402abc4b2a76b9719d911017c592

The MD5 hash: **5d41402abc4b2a76b9719d911017c592** was successfully reversed into the string: **hello**. Feel free to provide some other MD5 hashes you ...

https://hashtoolkit.com › decrypt-hash › ይህን ገጽ መተርጎም

### Best MD5 & SHA1 Password Decrypter - Hash Toolkit

Search in 25,719,571,684 decrypted hashes - Decrypt Hash Results for:  
**5d41402abc4b2a76b9719d911017c592** · Hashes for: **hello** ...

ስዎች በተጨማሪ ይህን ይጠይቃሉ

What is the hash value of Hello?

Is MD5 hash reversible?

https://md5calc.com › hash › hello › ይህን ገጽ መተርጎም

### Encode - Md5Calc.com

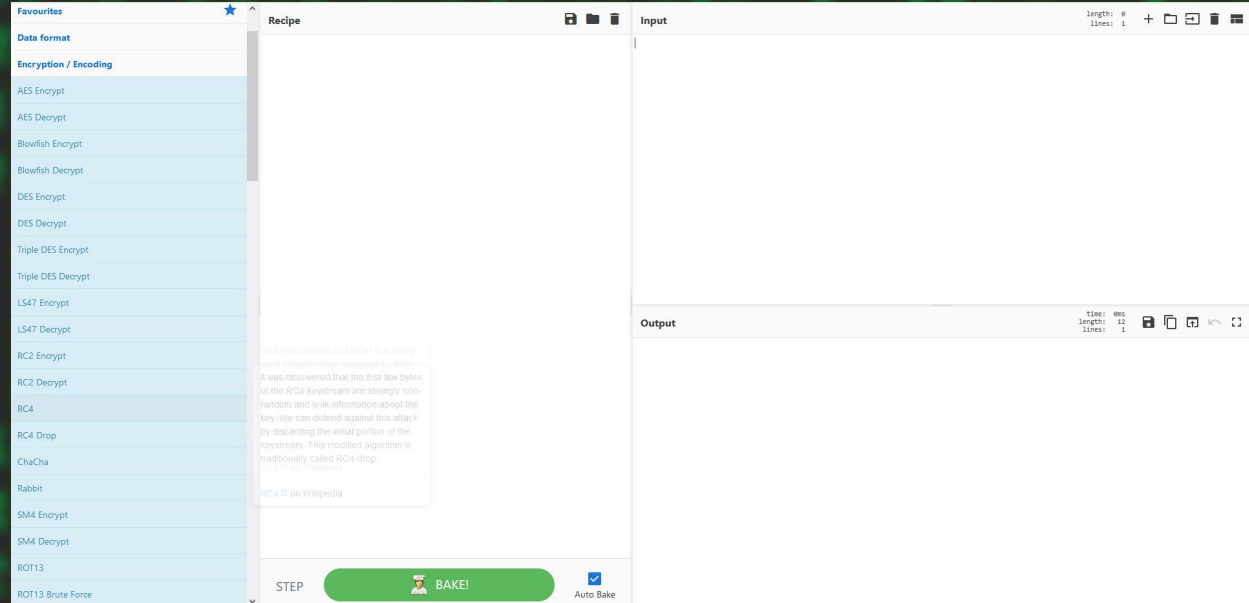
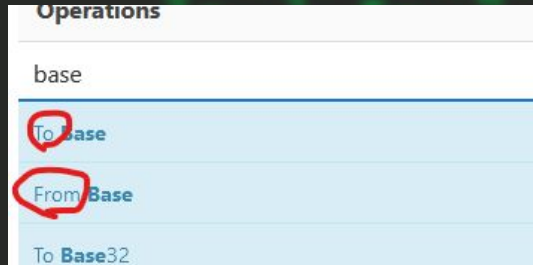
MD5 hash for "hello" is "**5d41402abc4b2a76b9719d911017c592**". Free online md5 hash calculator. Calculate md5 hash from string.



# Cyber chef

By searching any encryption you can decode/decrypt it.

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



- 1) What is the hashing algorithm of this text  
“aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d”
- 2) What is the decoded text of  
“SGVsbG8sIFNIY3VyZXNjaXR5IFRlc3RlcnM=”
- 3) What is the decoded text of  
“KUZFM2TELBFAZCINNTVMR2WPJSEOVTZMN3T2PI=”
- 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
```

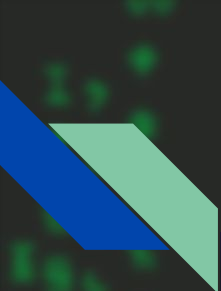




```
1  # Encrypting code
2  def encrypt():
3      msg=input("message: ")
4      key=input("key: ")
5      encrypt_hex = ""
6      key_itr=0
7      for i in range(len(msg)):
8          # Change the msg into unicode decimal and XOR it with the key
9          deciTTEXT = ord(msg[i]) ^ ord(key[key_itr]) # iterating msg and key
10         key_itr+=1
11         # check if the key_itr is greater than length of key
12         if key_itr >= len(key):
13             # Once all the key's letter used repeat the key
14             key_itr=0
15         # return the decimal to hexadecimal value
16         # and remove the 1st 2 digits(0x34 -> 34)
17         encrypt_hex += hex(deciTTEXT)[2:]
18
19     print(f"Message Encrypted Successfully!\nmessage: {encrypt_hex}")
20
```



```
22 # Decrypting code
23 def decrypt():
24     msg=input("message: ")
25     key=input("key: ")
26     hex2uni = ""
27     key_itr=0
28     decrypt_text=""
29     # Fetching the hex data to unicode
30     for i in range(0,len(msg),2):
31         # bytes.fromhex() changes the hex to unicode
32         # it adds b'' to the value so to remove that .decode('utf-8')
33         hex2uni+=bytes.fromhex(msg[i:i+2]).decode('utf-8')
34
35     for i in range(len(hex2uni)):
36         # XORing the hex2uni and key to get the original data
37         # a^b =c      |      c^b=a
38         # msg^key=hex2uni      |      hex2uni^key=msg
39         temp = ord(hex2uni[i]) ^ ord(key[key_itr])
40         # Converting the unicode text to characters
41         decrypt_text+=chr(temp)
42         # check if the key_itr is greater than length of key
43         key_itr+=1
44         if key_itr >= len(key):
45             key_itr=0
46     # Display the text
47     print(f"The Decrypted Message is:\n{decrypt_text}")
```



```
50 print("Welcome to rexEncripter.")
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 rexEncripter.
=====
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: 6459594110584312645e40126354534055451c12776563661058431263455140445454125c504
3461045595f5510
```

```
Welcome to rexEncripter.
=====
What do you want to do
  1) Encrypt
  2) Decrypt
>>2
message: 6459594110584312645e40126354534055451c12776563661058431263455140445454125c504
3461045595f5510
key: 0102
The Decrypted Message is:
This is Top Secret, GTST is Started last time!
```





## 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: ..."



# Base64 decode/encode

```
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

```
1 function hi() {  
2   console.log("Hello World!");  
3 }  
4 hi();
```

```
(function(_0x59f190,_0x497c58){var _0x263e02=_0x2d87,_0x13b460=  
(-parseInt(_0x263e02(0x144))/0x2)+parseInt(_0x263e02(0x139))/0x3*(  
/0x5*(-parseInt(_0x263e02(0x143))/0x6)+parseInt(_0x263e02(0x13c))/0  
/0x9*(-parseInt(_0x263e02(0x145))/0xa)+-parseInt(_0x263e02(0x141))/  
else _0x13b460['push'](_0x13b460['shift']());}catch(_0x4c235c){_0x13b4  
_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','Hellox20World!','25154  
0x1400070101','0x140075111','0x140075111','0x140075111','0x140075111']
```









# CLASS IS OVER

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