

Tarea: Reto Hackrocks

Asignatura: Criptografía

Alumno: Jaime Navarro

He empezado metiendo el texto en una variable de un código en Python para jugar con este archivo. Según la primera pista que pedí en Hackrocks, cada línea es un cifrado, pero no me cuadra porque no existe ningún cifrado hash que devuelva una cadena de texto de 188 caracteres. He contado los caracteres de cada línea, todos son de 188.

```
75 5bd17e4fbfe7de7c3db246430a192d6dbb2dc3e623ca604390e23fc20284d24a4f8c65ef32d35c8b552f4e21b03dc164d0
76 03af25070c32f3e3e462f5496de279573a2d22776db855b21c14f4e0006a0a0b09061d6ac235636394d3fd2afc16d4561d
77 f6ba54c3bbdbcb0397a1323fff2d50c92a9ea5330ca2bf00aeca9a2e3d74ce3315bc67a1f54afe8134f641ad314935b8f7
78 771c446373260cd8804332b3aec38e3fca5c8351870d72b0f0d0ff8e09ccb702a43ce523bf30a7bc502008afb78a80d303
79 c0a99cb4f1cc56e75dfa0bb3f1698f67b65574d651e7d5a867b58f5f97efd951d439e3d86e1aabee237c8fbee94c4c0423
80 54723d2276422728f80b61a818e8cd408e123d8fe45a376bbcb8570bdbcd29810ef553b4cf3a42e39c1c050f1e0ceef8b1
81 c45fdca75cb6ffd4fcfef7be5f125ca063d2317e714c487b02dc64308964bbfab5115f13d5e6ecb705fe7ca112327f9520
82 59b8e21ffa7e4a7e461ed316edef85ee9ff2d50496a0fd88194f1bac127a30107ec5a99ec89ac6dc3187fb8e6abd81f7c4
83 bffbada814877c201749151a218216e1fb9c293d0261234f854e1b80ee02e17d4015ed24dec5da3b83af09fb4ebb621651
84 9989825a3f46082dc0e412b8ce7c53140f451d8b387077bb7d5cbe0e4fc98799748af7dfa9056b3122e66d420616b80be2
85 12eab092b1d73790522767deaac15d0043f659ee74ee9f857b2406c795ec1683e9a36040974798438c039ed51617b0abd2
86 e8b371110c49559b3c7ba4ad8b4019ae00ba60173f326a5eaf3511077f2d98ad78fe71072e66431cddb42e5eaf562a196
87 c2ddfc20f4415facf41728810fa37ced5e2a28be0c258167aa24e2096951b4198ab6d0c1a3e1ba70c325fdbcb8650a979d0
88 a7a436f1bbe3b073b48c7dc11dc6393c1e6d7bfe2a1915b560e4c557bdc4c70a92737f7c0e511dc0261f37376aabed66a6
89 011d180a9bd3a4939a8d12e4e88fd20c2a9ac50ff3e934e646c8a704a77fbb02483589b8daa84e8b67050fc0476ebfb0a
90 1d3c797aec5e782038e18c0b4d0561daf97145b9d5fcb5fe4052cb45715b949849eb0641b646e8c4af5d027fc0c113f79f
91 da31f23c8342d960f5448f126bc601466d3146f3f4fd5e750a149783dad97ffa999224bae388c02eeaaaf3c2536c010d02
92 2497d63a0b338e99d2eabab6c3ae0d91bad3ab14a7ab8637bb6daaddfec566afdb5a11a8c6b167878b7f43db2c4e138886
93 c6a388d0ee163e5b1ea559bf825295d4e5d996f1fa58ff19c6a93d0c017b293b73f761f89d3a489761673b7f4df6023c86
94 058e0ee31fe6a2575591ef9951c769ed3f07cb09442271e1b49507b43167ba2f0750236d8130cb5558d1889aade2286c61
95 3cfa0ad2dfdcd62c534803415a91d60716b729da9dd118044cae11f3424b8addf6bd609492d0d034eb168e9d8c267dbed
96 d6088c3fd09df00da290b9ca8fce47fcb557eacf29f0a578ab65dc04bd8ba00ee3aab68af048fc72ade4785909c6b83090
97 8efc29d76aab1fe7fe5f1ebf82adeee02173cd9dc7d46cd14a394d4be271cfe01d4d33d8c79c7f0c132cd49ac85b8ed0fc
98 354c2b1e657065dc1e94c6c57e9f1ff28e2b3b301e5d390d36d4de87e6bd97a70efbe066ce4fc25275db3f7e02ba02cbb4
99 1a555e7825f92e9adeaef214c7e77a16391a00f634b814b2830a8512b6642084b3466a0eb0e2eea28780772be0c6d48612
100 d4d868adea901288bedd7be625a71e7ba947a4317ca026ad2becfb61084ce4fa6baf6902ae7514107c773188f036973bb4
101 print(len(a),"\tletras en total")
102 lista = a.split()
103 print(type(lista))
104 for line in range(len(lista)):
105     print("longitud de la línea ",line,"\t es\t",len(lista[line]))
106 # cómo leer un str de varias líneas, guardándolo en una lista a elemento por línea, en python?
107 """import os
108 os.system("whoami")
109 for line in range(len(lista)):
110     os.system("hash-identfier"+" "+lista[line])"""
```

ufv.hackrocks.com/challenges/start/hidden-patterns

Puedes utilizar:

**Pistas**

**1 de 3**

Disponibles

Usar pista

**Mentoría**

**1 de 1**

Disponibles

Solicitar

Al resolver este reto, obtendrás **43** puntos

**Pista nº 1**

Una de las líneas, con seguridad, no contiene información aleatoria, y ha sido ocultada entre las demás para dificultar su detección.

**Pista nº 2**

Sabemos que, en otras ocasiones, este grupo ha utilizado un cifrador sencillo, basado en una simple función XOR, con clave de un único byte.

Cabe decir que el código comentado fue un intento de ejecutarlo en Linux, para usar "hash-identifier", pero no resultó. Luego para Windows tocó comentarlo.

SALIDA      TERMINAL      CONSOLA DE DEPURACIÓN      PROBLEMAS

```
_nuevo/patrones_ocultos/descifrando_output.py"
```

```
18899  letras en total
```

```
<class 'list'>
```

```
longitud de la línea 0            es        188
```

```
longitud de la línea 1            es        188
```

```
longitud de la línea 2            es        188
```

```
longitud de la línea 3            es        188
```

```
longitud de la línea 4            es        188
```

```
longitud de la línea 5            es        188
```

```
longitud de la línea 6            es        188
```

SALIDA      TERMINAL      CONSOLA DE DEPURACIÓN      PROBLEMAS

```
longitud de la línea 91           es       188
```

```
longitud de la línea 92           es       188
```

```
longitud de la línea 93           es       188
```

```
longitud de la línea 94           es       188
```

```
longitud de la línea 95           es       188
```

```
longitud de la línea 96           es       188
```

```
longitud de la línea 97           es       188
```

```
longitud de la línea 98           es       188
```

```
longitud de la línea 99           es       188
```

```
PS C:\Users\████████████████████\Desktop\████████████████████\TERCERO_3º> █
```

Se puede ver que todas las líneas tienen la misma longitud.

Con la ayuda de ChatGPT, busqué los hashes que se basan en la operación XOR, dado que recuerdo que algo dimos de un cifrado relacionado con esa operación, pero no cuál, y tengo que darme prisa. Estos me han dicho:

chat.openai.com/chat/95fd53-1d96-4a54-ba87-56e86144a99f

JU

dime todos los posibles funciones de cifrado hash si tenemos que cuenta los que se basan en una operación XOR



Es importante tener en cuenta que el XOR no es un algoritmo de cifrado hash, sino un operador lógico que se utiliza en algunos algoritmos de cifrado hash para combinar datos de entrada. Por lo tanto, no existen funciones de cifrado hash que se basen exclusivamente en la operación XOR.



Sin embargo, existen varios algoritmos de cifrado que utilizan la operación XOR como parte de su proceso de cifrado. Algunos ejemplos incluyen:

1. RC4: es un algoritmo de cifrado de flujo que utiliza la operación XOR para combinar datos de entrada con una clave secreta para generar una secuencia de cifrado. La secuencia de cifrado se utiliza para cifrar y descifrar los datos de entrada.
2. Salsa20: es un algoritmo de cifrado de flujo que utiliza la operación XOR para combinar datos de entrada con una clave secreta y un vector de inicialización para generar una secuencia de cifrado. La secuencia de cifrado se utiliza para cifrar y descifrar los datos de entrada.
3. TEA (Tiny Encryption Algorithm): es un algoritmo de cifrado de bloque que utiliza la operación XOR para combinar datos de entrada con una clave secreta en múltiples rondas para generar una salida cifrada.
4. XTEA (Extended TEA): es una variante del algoritmo TEA que utiliza la operación XOR y otras operaciones aritméticas para mejorar su seguridad.

Ten en cuenta que estos algoritmos no son algoritmos de cifrado hash, sino algoritmos de cifrado simétrico que utilizan la operación XOR como parte de su proceso de cifrado. Los algoritmos de cifrado hash, por otro lado, son algoritmos unidireccionales que se utilizan para generar un valor resumen de un conjunto de datos de entrada.

Fuente: <https://chat.openai.com>

## Search for a tool

★ SEARCH A TOOL ON DCode BY KEYWORDS:

e.g. type 'random'

★ BROWSE THE FULL DCode TOOLS' LIST

## Results

354C2B1E6570...ADC

↑↓	↑↓
[163]	3CF99FB43F0CCECBDBD48B302CDF3D427BAB82 1634F5B4B083D51DBE0A63775BCBDE7BCA703A 6F56CEC0FFCA242602C3CE34109EE67AE19A4C 9E37521F49317E176FBE321A4E8DA202447165 7080385447951207CA3FCEA6653EEB4D8944
[25]	34498D4BEE3566AECF3AC70215974458308075 61744202AD8E4A123EBF671E71394F12573E9C 4A91EA76EE90143EC3C206242ACFFOCEC65F6A 307ACB7EDBCB57736A0EBD15D380B4B5F43612 8B8B724D3FDC203B84DBDBCECCOF34BF99FC
[93]	32DA126ECF5B4592223F3EA680CB1E47873D76 077E94463D794DCEA0FB3B8EB2CF4F3E65E654 8ECC153075CEBB4CB5CC0AB106723ADA7453C4 41CC67EAF700A099BEEB8FD2D9300E345F1331 342B8AFF1F9B70CECD9413D2647A6A18244B
[109]	30440032468EC29B895103DB7EC3DF4676B713 5D855F14ADEAFF1F3B748F2E350A54C7307BF8 366CE49E8B3AEA1F4EC73D67F5DE8A34356EC2 401AB8629FBC0E1EAC392A2E4CC0CA1EC29472 7F4130DBCE703497CB4972CD86A16502B4DB
[129]	347B2F64848C723ACEC716190E5BCEC4959B36 321C576B4D13D230FC7ADCCF2330907E4E0B9F C642EB3A1ABF7ACD62CDC45F8DA07401F5F1EE DEB6DF3E3BEE3AD54E3446C03720B04A4513BE 42528297877FB9617FAE05083AAB846B19C4
[49]	3C3BBEDF249FB282640B75BB6A311FA0164F82 FDCACEE26D75A934441B3FCA57859E84DD3550 F3771E67CD8B36D770C63E414F4FDF137AC2EE 7A743B9BD283CE0E91145AE9CC8068C69C3248 CEFE2B8A76204932373CC0A40E541A4E0005
[139]	35000E4A145E04A0CC37323940267ABB2EFEC8 423C96CB608CC9EA54119E0EC382DB9B347A7E E2CA731FDF4F414E36C077D638BDC76E1773F0 553DD48E95875ACF3B144439A57D62EECACDF2 8F4610AF113A6BB57B046282BF942FDEBB3C

## SKIP CIPHER

Cryptography › Transposition Cipher › Skip Cipher

## SKIP DECODER

★ SKIP CIPHERTEXT (?)

```
354c2b1e657065dc1e94c6c57e9f1ff28e2b3b301e5d390d36d4de87e6b
d97a70efbe066ce4fc25275db3f7e02ba02cbb448baca743213c48d5ee0
6b3f4fcabb8431f3afc44e7039429f7b0e793080471ac78ca13433249a4
a52c6611adc
```

★ IGNORE PUNCTUATION (AND SPACE, ETC.) ☒

## AUTOMATIC SKIP PARAMETER FINDER (BRUTEFORCE)

▶ DECRYPT

## MANUAL PARAMETERS

★ SKIP SIZE 17

★ INITIAL POSITION (START = 1) 1

▶ DECRYPT

See also: [Letters Extraction](#) — [Affine Cipher](#)

## SKIP ENCODER

★ SKIP PLAINTEXT (?)

dCode skip

★ SKIP SIZE 17

★ INITIAL POSITION (START = 1) 1

★ IGNORE PUNCTUATION (AND SPACE, ETC.) ☒

▶ ENCRYPT

## EXTRACT EVERY NTH CHARACTERS

▶ Go to: [Letters Extraction](#)

## Answers to Questions (FAQ)

## What is the Skip cipher? (Definition)

The skip cipher (or jump cipher) is a **transposition** cipher which reorders the letters of a message by **extracting a letter** every  $n$  characters (by jumping  $n$  characters or skipping  $n - 1$ ).

## How to encrypt using Skip cipher?

Set the size of the skip  $s$ , extract the first letter of the message, then make jumps of  $s$  characters. When the end of the message is reached, go back at the beginning (loop).

JU

dime algoritmos de cifrado hash cuya salida sea una cadena de texto de longitud fija igual o superior a 100 caracteres, que además usen la operación XOR de alguna forma



Uno de los algoritmos de cifrado hash que cumple con estas condiciones es Skein, que utiliza la operación XOR en su proceso de cifrado y devuelve una salida de longitud fija de hasta 1024 bits (128 bytes). Otro algoritmo es BLAKE2, que también utiliza la operación XOR y devuelve una salida de longitud fija de hasta 512 bits (64 bytes). Por último, podemos mencionar Whirlpool, que utiliza la operación XOR en su proceso de cifrado y devuelve una salida de longitud fija de 512 bits (64 bytes).



JU

dime algoritmos de cifrado hash cuya salida sea una cadena de texto de longitud fija igual o superior a 180 caracteres, que además usen la operación XOR de alguna forma



No hay muchos algoritmos de cifrado hash cuya salida tenga una longitud fija igual o superior a 180 caracteres y que utilicen la operación XOR. Sin embargo, aquí te presento algunos ejemplos:



1. Skein: Es un algoritmo de cifrado hash que utiliza la operación XOR de forma intensa. Su salida es de longitud variable, pero puede ser configurada para producir una salida de 224, 256, 384, o 512 bits, lo que equivale a 28, 32, 48, o 64 bytes respectivamente.
2. Blake2: Es otro algoritmo de cifrado hash que utiliza la operación XOR para generar su salida. Su salida es de longitud variable, pero puede ser configurada para producir una salida de hasta 64 bytes.
3. Hamsi: Es un algoritmo de cifrado hash que utiliza la operación XOR en su diseño. Su salida es de longitud fija y puede ser configurada para producir una salida de 256, 384 o 512 bits, lo que equivale a 32, 48 o 64 bytes respectivamente.

Es importante tener en cuenta que la operación XOR es una operación comúnmente utilizada en la construcción de funciones hash, pero no es el único factor que determina la seguridad y calidad de un algoritmo de cifrado hash.

JU

dime algoritmos de cifrado hash cuya salida sea una cadena de texto de longitud igual o superior a 180 caracteres, que además usen la operación XOR de alguna forma



No hay algoritmos de cifrado hash conocidos que produzcan una salida de longitud fija igual o superior a 180 caracteres y que usen la operación XOR de alguna forma. Los algoritmos de cifrado hash conocidos con salidas de longitud fija y superior a 180 caracteres suelen utilizar otras operaciones criptográficas, como la compresión, la permutación o la sustitución.

Me desespero, pido la tercera y última posible pista:

ufv.hackrocks.com/challenges/start/hidden-patterns

0 de 3  
Disponibles

No hay más pistas disponibles

1 de 1  
Disponibles

Solicitar

Al resolver este reto, obtendrás 37 puntos

**Pista nº 1**  
Una de las líneas, con seguridad, no contiene información aleatoria, y ha sido ocultada entre las demás para dificultar su detección.

**Pista nº 2**  
Sabemos que, en otras ocasiones, este grupo ha utilizado un cifrador sencillo, basado en una simple función XOR, con clave de un único byte.

**Pista nº 3**  
Hemos descubierto que la clave es el byte '36' (valor numérico, no cadena).



Pues nada, a probar suerte en <https://dcode.fr> .

Si pruebo a meter en "hash identifier" cualquier línea, me devuelve siempre los mismos 3 posibles:

The screenshot shows the 'HASH IDENTIFIER' tool on the dcode.fr website. The interface is divided into two main sections. On the left, there is a 'Search for a tool' section with a search bar containing 'e.g. type 'random'' and a link to 'BROWSE THE FULL DCODE TOOLS' LIST'. Below this is a 'Results' section showing a list of hash algorithms: 'Cisco Type 7', 'BigCrypt', and 'Umbraco HMAC-SHA1'. On the right, the 'HASH IDENTIFIER' tool is active, displaying a 'HASH/FINGERPRINT' input field with the value 'fd3dfb059e998df6961bd0fbf9c97bb48a2e78ea57a435f6f...'. Below the input field is an 'IDENTIFY' button. The right section also includes a 'See also: Cipher Identifier' link and an 'Answers to Questions (FAQ)' section with two sub-sections: 'What is a Hash identifier? (Definition)' and 'How to use the Hash identifier?'. The 'Definition' section explains that a hash is a number stored in hexadecimal format, and the 'How to use' section instructs users to provide a character string that serves as a fingerprint.

**Search for a tool**

★ SEARCH A TOOL ON DCode BY KEYWORDS:  
e.g. type 'random'

★ BROWSE THE FULL DCODE TOOLS' LIST

**Results**

The following hash algorithms generate a fingerprint similar to `fd3dfb059e99...1eb`

↑↓

- Cisco Type 7
- BigCrypt
- Umbraco HMAC-SHA1

Hash Identifier - dCode  
Tag(s) : Modern Cryptography, dCode, Informatics

Share

## HASH IDENTIFIER

Cryptography > Modern Cryptography > Hash Identifier

### HASH IDENTIFIER

★ HASH/FINGERPRINT

fd3dfb059e998df6961bd0fbf9c97bb48a2e78ea57a435f6f...

► IDENTIFY

See also: [Cipher Identifier](#)

### Answers to Questions (FAQ)

#### What is a Hash identifier? (Definition)

There are hundreds of hashing algorithms, the majority return a **hash** as a number, usually stored in **hexadecimal** format. But the length of this number, some additional characters or simply the format of the final string allows to recognize what type of algorithm was used.

#### How to use the Hash identifier?

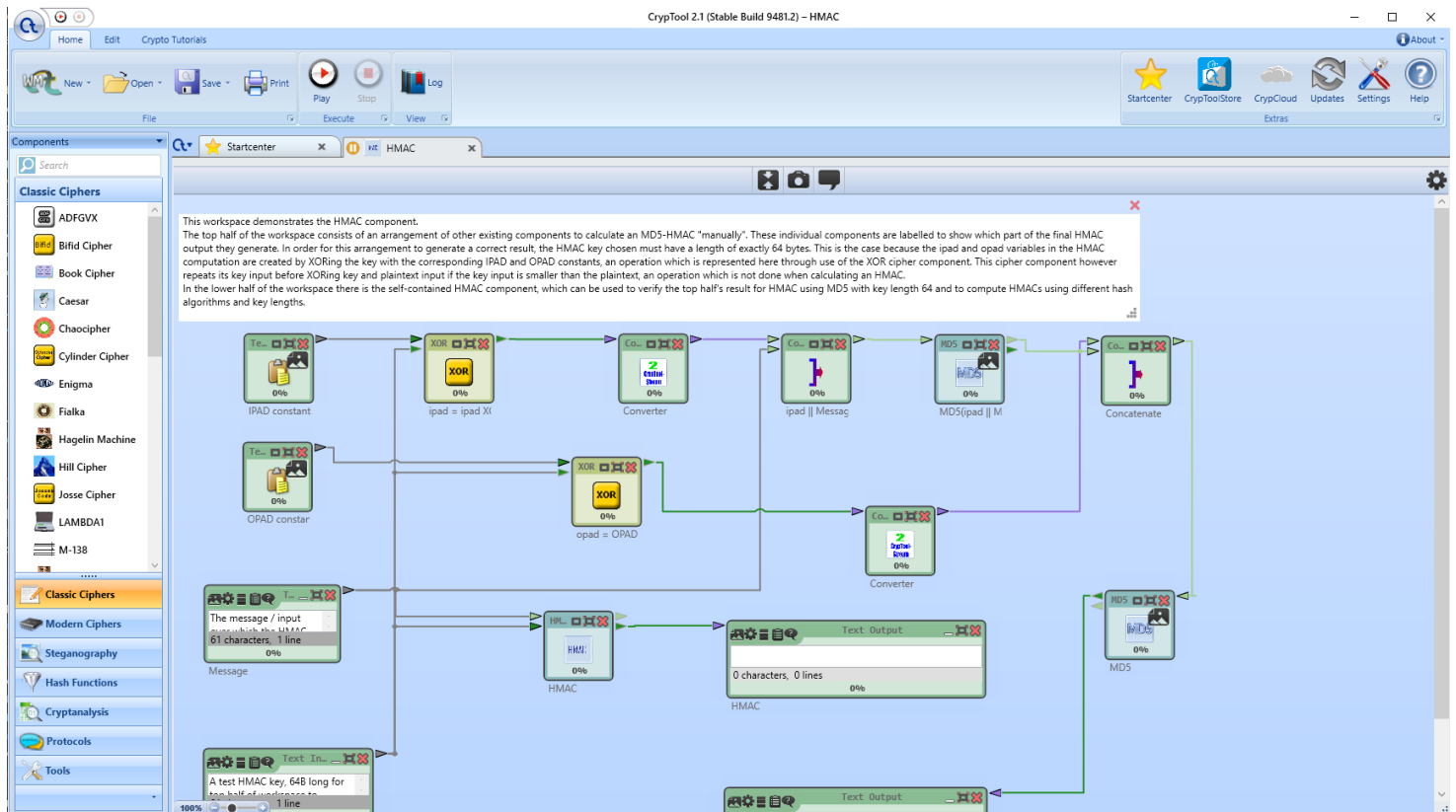
Indicate the character string that serves as a fingerprint. Take care not to add unnecessary elements, such as a space or a stop at the end, as these irrelevant characters could confuse the detector.

No tengo idea, probaré con los 3.

No los encuentro...



He probado con Cryptool... pero algo me dice que para esta entrega no me da tiempo a usarlo, tendré que familiarizarme con esta herramienta.



# CRYPTOOL 2

Welcome to Cryptool 2. There are two ways for a quick start: Click on the wizard button in the section "Main Functions" to get a guided tour, or load one of the pre-defined workflows in the section "Templates" (double click to open)

### Main functions

- Create a new workspace with the graphical editor
- Use the wizard to easily try some Cryptool 2 features

### External resources

- View a tutorial video about Cryptool 2 on YouTube

### Templates (double click to open)

Search 266

- Modern
  - Hash Functions
    - BLAKE
    - CRC Null Problems
    - CRC
    - Dictionary Attack on a Password Hash Value
    - Grøstl
    - HMAC
    - ImageHash
    - ImageHash - Smooth Comparison