Introduction to cryptography with cryptopals for the scared programmer

Cryptopals Introduction

f3rn0s

August 13, 2019

Introduction

Who Am I?

Background
University Student

Overview

What this talk is
Get some understanding of the more complicated topics in solving this challenge, and then solve the challenges with assistance.

What this talk isn't
A look at code solutions to challenges from cryptopals.

Content

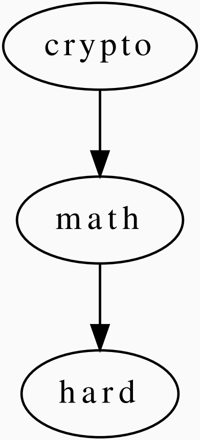
- Brief Cryptography Introduction
- Single Char XOR
  - What is XOR
  - XOR of Binary values
  - XOR of Integers
  - XOR of Characters
  - XORing strings in Python3
- Repeating Key XOR
  - Hamming Distance
  - Transposition
- Block Encryption
  - Block Ciphers
  - Discovering appended secrets
- Mersenne Twister (MT19937)
  - Implementation
  - Bad Seeds
    - Using Unix Epoch Time
    - Using a 16 bit integer
  - How to attack
- Choose one of the three topics to solve a challenge for

Cryptography

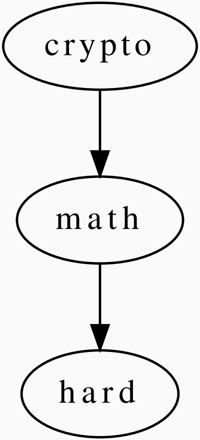
What is it?

Any of various mathematical techniques for encrypting and decrypting data in order to keep it private when transmitted or stored.

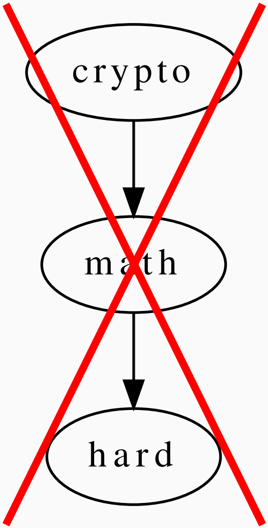
Math?!



Math?!



Math?!



How Much Math Do I Need To Know?

To Quote Cryptopals:

"If you have any trouble with the math in these problems, you should be able to find a local 9th grader to help you out. It turns out that many modern crypto attacks don't involve much hard math."

XOR

What is XOR

Definition

XOR is a boolean operator that outputs true if both inputs are different.

It is usually defined by a $\oplus$ in maths, but a $\wedge$ in programming languages.

Useful Property

$$a \wedge b = c$$

$$a \wedge c = b$$

$$b \wedge c = a$$

Binary

- Goes from Right to Left

$45 =$	0	0	1	0	1	1	0	1
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Table 1: Example Binary value

XOR Binary

XORing two binary values is simply using XOR for each bit, so the first bit of each value XORed against each other, then the second etc.

0	0	1	0	1	1	1	0
			$\wedge$				
1	1	1	1	0	0	0	0
			$=$				
1	1	0	1	1	1	1	0

Table 2: Example Binary XOR

XOR Integers

1. Convert Integer to Binary
2. XOR Binary values
3. Convert Binary to Integer

Note
Many programming languages allow for "bitwise" operators that include bitwise XOR that can be used on integers; for example: C, C++, Java, Python, Rust, Ruby, F#, Haskell, pretty much everything tbh.

XOR Integers Examples

$$20 \wedge 15 =?$$

$$20 = 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 = 10100$$

$$15 = 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 01111$$

1	0	1	0	0
		$\wedge$		
0	1	1	1	1
		$=$		
1	1	0	1	1

$$11011 = 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 27$$

$$\therefore 20 \wedge 15 = 27$$

XOR Integers Examples

Proving the property that if $a \wedge b = c$ then $b \wedge c = a$

$$27 \wedge 15 = ?$$

$$27 = 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 11011$$

$$15 = 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 01111$$

1	1	0	1	1
		$\wedge$		
0	1	1	1	1
		$=$		
1	0	1	0	0

$$10100 = 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 = 20$$

$$\therefore 27 \wedge 15 = 20$$

XOR Characters pt.1

	2	3	4	5	6	7
0:		0	@	P	`	p
1:	!	1	A	Q	a	q
2:	"	2	B	R	b	r
3:	#	3	C	S	c	s
4:	$	4	D	T	d	t
5:	%	5	E	U	e	u
6:	&	6	F	V	f	v
7:	'	7	G	W	g	w
8:	(	8	H	X	h	x
9:	)	9	I	Y	i	y
A:	*	:	J	Z	j	z
B:	+	;	K	[	k	{
C:	,	<	L	\	l	|
D:	-	=	M	]	m	}
E:	.	>	N	^	n	~
F:	/	?	O	_	o	DEL

	30	40	50	60	70	80	90	100	110	120
0:		(	2	<	F	P	Z	d	n	x
1:		)	3	=	G	Q	[	e	o	y
2:		*	4	>	H	R	\	f	p	z
3:	!	+	5	?	I	S	]	g	q	{
4:	"	,	6	@	J	T	^	h	r	|
5:	#	-	7	A	K	U	_	i	s	}
6:	$	.	8	B	L	V	`	j	t	~
7:	%	/	9	C	M	W	a	k	u	DEL
8:	&	0	:	D	N	X	b	l	v	
9:	'	1	;	E	O	Y	c	m	w	

Figure 1: ASCII Compact Table (Man Page)

XOR Characters pt.2

1. Convert Character to associated Integer (Uses ASCII Values)
2. Convert Integer to binary
3. XOR Binary
4. Convert Binary to Integer
5. Convert Integer to Character (Using ASCII Values)

Note
In my experience, hardly any languages have a way to XOR two characters directly, so the conversion to/from an integer is necessary.

XORing Strings pt.1

Start:

A		s	t	r	i	n	g
$\wedge$							
A	A	A	A	A	A	A	A

Conversion:

65	32	115	116	114	105	110	103
$\wedge$							
65	65	65	65	65	65	65	65
$=$							
0	97	50	53	51	40	47	38

XORing Strings pt.2

0	97	50	53	51	40	47	38
As String							
\00	a	2	5	3	/	(	&

Table 3: Resulting String

Problem: Null character at start will break your string.

Solution: Encode the output to some other format, such as base32, base64 etc.

Breaking Single Char XOR

What is Single Character XOR?

Select some arbitrary Character (or Integer) and use it to encode a string.

A		s	t	r	i	n	g
^							
A	A	A	A	A	A	A	A
=							
\00	a	2	5	3	(	/	&

Table 4: 'Encrypting' with XOR

\00	a	2	5	3	(	/	&
^							
A	A	A	A	A	A	A	A
=							
A		s	t	r	i	n	g

Table 5: 'Decrypting' with XOR

Example XOR for Single Character in Python3

```
def xor(buf1, charbuf):
    return bytes([
            (buf1[i] ^ charbuf[0]) for i in range(0, len(buf1))]
          )

xor(b'A string', b'A')
# Outputs: b'\x00a253(/&'
```

English Frequency Analysis

Letter	Relative frequency in the English language
e	12.702%
t	9.056%
a	8.167%
o	7.507%
i	6.966%
n	6.749%
s	6.327%
h	6.094%
r	5.987%
d	4.253%
l	4.025%
c	2.782%
u	2.758%
m	2.406%
w	2.360%
f	2.228%
g	2.015%
y	1.974%
p	1.929%
b	1.492%
v	0.978%
k	0.772%
j	0.153%
x	0.150%
q	0.095%
z	0.074%

Figure 2: English Frequency (Wikipedia)

English Frequency in Python

Example dictionary for English frequency in Python.

```
freq = {'e': 12.702, 't': 9.056, 'a': 8.167, 'o': 7.507,
        'i': 6.966, 'n': 6.749, 's': 6.327, 'h': 6.094,
        'r': 5.987, 'd': 4.253, 'l': 4.025, 'c': 2.782,
        'u': 2.758, 'm': 2.406, 'w': 2.360, 'f': 2.228,
        'g': 2.015, 'y': 1.974, 'p': 1.929, 'b': 1.492,
        'v': 0.978, 'k': 0.772, 'j': 0.153, 'x': 0.150,
        'q': 0.095, 'z': 0.074, ' ': 13.000}
```

Breaking Single Character XOR

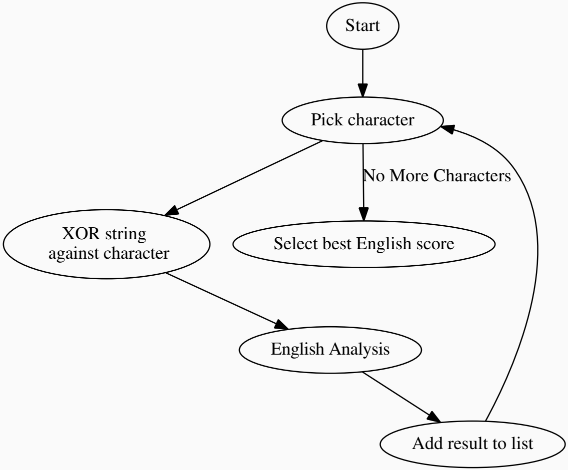


Figure 3: Logical flow of breaking single character XOR

First Challenge

`https://cryptopals.com/sets/1/challenges/3`

or

`singlexortemplate.py` in
`https://github.com/f3rn0s/cryptopals-talk-starters`

Repeating Key XOR (Viegnere)

Overview of steps on Cryptopals

1. Calculate Key Size using hamming distance
2. Break string into chunks of key size
3. Transpose the chunks
4. Solve each transposed chunk as a single char xor
5. Use solved chunks to build the repeating key

Note
If you can do this one, you're probably just fine up to Set 6. ~Cryptopals

Working out key size

1. Guess key sizes between 2 — 40
2. Calculate the hamming distance between the bits of the first two guessed key size chunks (or more for better accuracy)
3. Divide the hamming distance by the guessed keysize
4. The key size with the smallest hamming distance is probably the key

Why does this work (Key size)

ASCII strings run from values 0–127 and as such do not use an 8th bit, therefore when the key size is correct the distance will be lower as:

A) the first bit will be 0 every time and

B) ASCII values are fairly similar.

Why does this work (Transpose)

Repeating key: "YELLOW SUBMARINE"

String: "A stringEveryoneEnglish sentence"

A		s	t	r	i	n	g	E	v	e	r	y	o	n	e
Y	E	L	L	O	W		S	U	B	M	A	R	I	N	E
E	n	g	l	i	s	h		s	e	n	t	e	n	c	e
Y	E	L	L	O	W		S	U	B	M	A	R	I	N	E

Transposed String

`AE`	Y
`n`	E
`sg`	L
`tl`	L
`ri`	O
`is`	W
`nh`	
`g`	S
`Es`	U
`ve`	B
`en`	M
`rt`	A
`ye`	R
`on`	I
`nc`	N
`ee`	E

Second Challenge

`https://cryptopals.com/sets/1/challenges/6`

or

`repeatingkeytemplate.py` in
`https://github.com/f3rn0s/cryptopals-talk-starters`

AES ECB

PKCS#7

Block ciphers used fixed-sized blocks, usually either 8 or 16 bytes, as such the key must be the size of the mode (e.g. AES-128 is 16 bytes) and the plain text must be a multiple of the block size (e.g. if the block size is 16 bytes, the plain text must be 16, 32, 48, 64 etc. bytes).

PKCS helps bypass the problem of needing specific block size plaintext by adding consistent padding to the input. PKCS7 will always add bytes to the input. It will add n bytes with a value of n, where n is the number of bytes needed to reach the block size.

Example: '`YELLOW SUBMAR\x03\x03\x03`'

Example: '`YELLOW SUBMARINE\x16\x16\x16\x16\x16\x16\x16\x16\x16\x16\x16\x16\x16\x16\x16\x16\x16`'

PKCS#7 is used in the challenge, but not used in the examples.

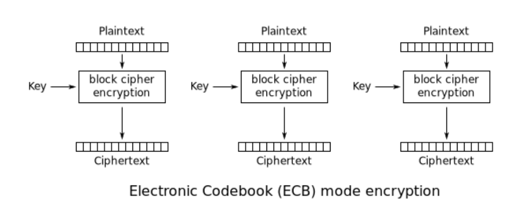


Figure 4: ECB Encryption

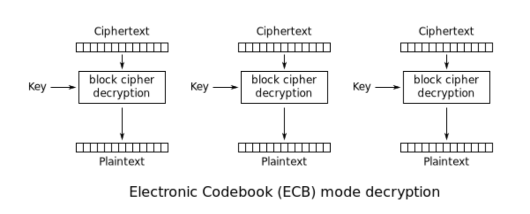


Figure 5: ECB Decryption

Problem with Block Ciphers

When encrypting a String with a key, each 16 byte block in that string will be encrypted individually, and if any block is repeated it will produce the same encrypted output.

e.g. AAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBAAAAAAAAAAAAAAAA

The A's will produce the same resultant block.

For example, if the key is 'YELLOW SUBMARINE'.

```
1st block:  \xe4@\x85\xfa+\xda3\xd8j\xa3@\xb4\xc1l\x05\xad
2nd block:  \xc4w\x81Ll\xd1-\x92"\xff?\x90\xe0\xe3@7
3rd block:  \xe4@\x85\xfa+\xda3\xd8j\xa3@\xb4\xc1l\x05\xad
```

Block 1 and 3 are the same.

Controlled plain text

```
AES-128-ECB(pkcs(your-string + unknown-string), random-key)
```

Can we extract the unknown string?

Third Challenge

`https://cryptopals.com/sets/2/challenges/14`

or

`ecbtemplate.py` in
`https://github.com/f3rn0s/cryptopals-talk-starters`

Mersenne Twister

Is It Secure?

Wikipedia
Is not cryptographically secure, unless the CryptMT variant (discussed below) is used. The reason is that observing a sufficient number of iterations (624 in the case of MT19937, since this is the size of the state vector from which future iterations are produced) allows one to predict all future iterations.

Note: Not the solution to the cryptopals' MTcipher challenges

MT19937 Variables

For a w-bit word length, the Mersenne Twister generates integers in the range $[0, 2^w - 1]$.

$$(w, n, m, r) = (32, 624, 397, 31)$$

$$a = \texttt{9908B0DF}_{16}$$

$$(u, d) = (11, \texttt{FFFFFFFF}_{16})$$

$$(s, b) = (7, \texttt{9D2C5680}_{16})$$

$$(t, c) = (15, \texttt{EFC60000}_{16})$$

$$l = 18$$

If you want to implement

https://en.wikipedia.org/wiki/MT19937

Seeding

Based on the initial seed the PRNG will generate the same values. e.g. If the PRNG is seeded with 0, then no matter how many times the program is run the 'random' values will always be the same.

Bad Seed — Time

Using the current epoch time is a *start*, but if someone knows when your application was started, they can easily countdown from the current time until they recover the seed.

e.g. If they seed it with 1565586530 (Example epoch time), and you try to break it two minutes later, you can simply try all 120 values decreasing from 1565586650.

Bad Seed — Small Number

Arbitrarily choosing a number that is less then 16 bits is a bad idea, it is not hard to brute force all 65536 (2^{16}) values.

It is even achievable to brute force up to all 19 bits 1048576 (2^{19}) in a reasonable amount of time.

Making a cipher out of a twister

~Cryptopals
You can create a trivial stream cipher out of any PRNG; use it to generate a sequence of 8 bit outputs and call those outputs a keystream. XOR each byte of plaintext with each successive byte of keystream.

Fourth Challenge

https://cryptopals.com/sets/3/challenges/24

or

mttemplate.py in https://github.com/f3rn0s/cryptopals-talk-starters against https://f3rn0s.pythonanywhere.com.

Questions?