Simple Crypto

ss8651twtw

$ whoami

- 林思辰
- 資工系大三
- BambooFox
- ss8651twtw / linsc04
- ss8651twtw@gmail.com

Outline

- Basis
- Classical cipher
- eXclusive OR
- Hash
- Rivest Shamir Adleman

Basis

String transform

- ASCII
  - BambooFox
- Hex
  - 0x42616d626f6f466f78
- Decimal
  - 122450548489718656888
- Binary
  - 1000010011000010110110101100010011011110110111101000110011
    01111101111000

String transform

- Python
  - 'BambooFox'.encode("hex")
  - '42616d626f6f466f78'.decode("hex")
  - int('0x42616d626f6f466f78', 16)
  - int('100001001100001011011010110001001101111011011110100011001101111011111000', 2)
  - hex(1224505484489718656888)
  - bin(1224505484489718656888)

String transform

- Practice
  - https://bamboofox.cs.nctu.edu.tw/admin/courses/4/challenges/73
  - https://bamboofox.cs.nctu.edu.tw/admin/courses/4/challenges/74
  - https://bamboofox.cs.nctu.edu.tw/admin/courses/4/challenges/75

Base64

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Base64

source ASCII (if <128)	M								a								n							
source octets	77 (0x4d)								97 (0x61)								110 (0x6e)							
Bit pattern	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
Index	19								22				5				46							
Base64-encoded	T								W				F				u							
encoded octets	84 (0x54)								87 (0x57)				70 (0x46)				117 (0x75)							

Base64

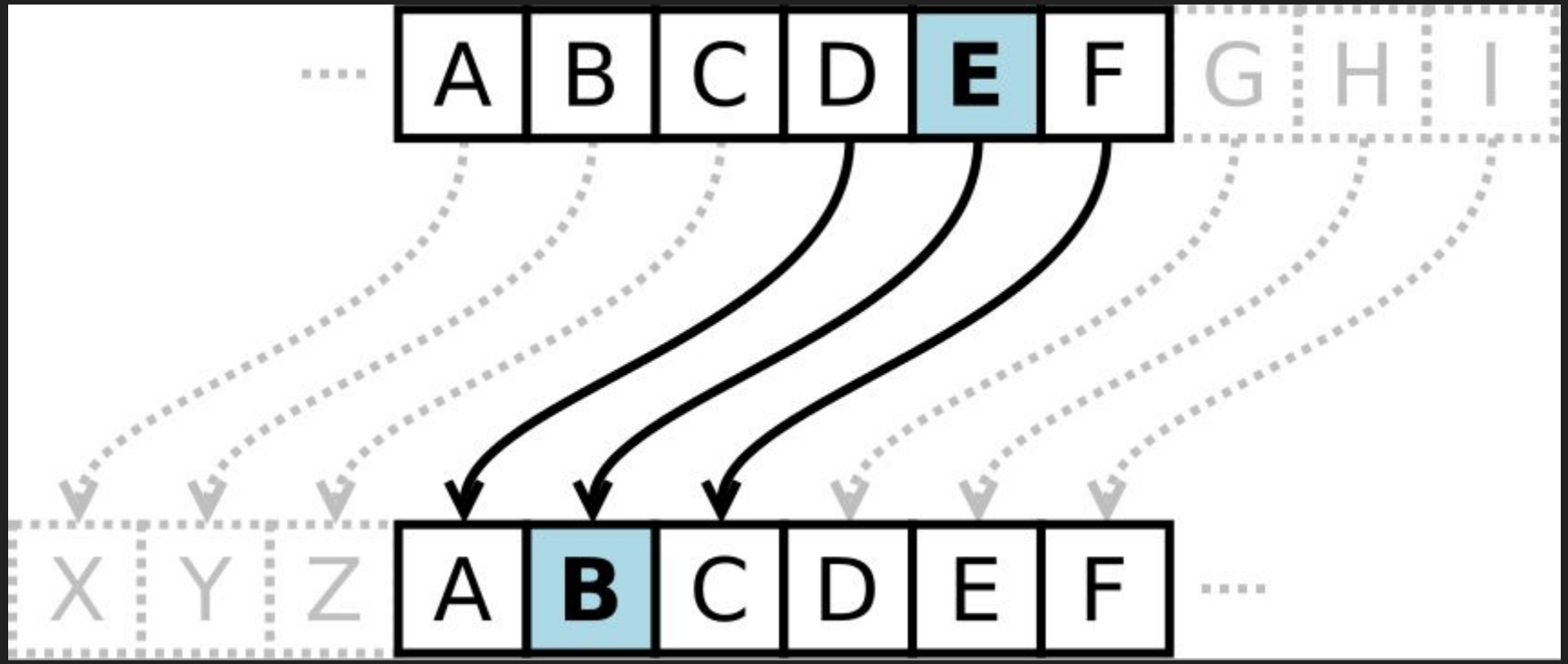
Text content	M																							
ASCII	77 (0x4d)								0 (0x00)								0 (0x00)							
Bit pattern	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index	19								16								0							
Base64-encoded	T								Q								=							

Base64

- Online tool
  - https://www.base64encode.org/
  - https://www.base64decode.org/
- Practice
  - https://bamboofox.cs.nctu.edu.tw/admin/courses/4/challenges/76

Classical cipher

Caesar cipher



Caesar cipher

- Encrypt

$$E_n(x) = (x + n)\ mod\ 26$$

- Decrypt

$$D_n(x) = (x - n)\ mod\ 26$$

Caesar cipher

- Online tool
  - https://planetcalc.com/1434/

Substitution cipher

- Plaintext

```
the quick brown fox jumps over the lazy dog
```

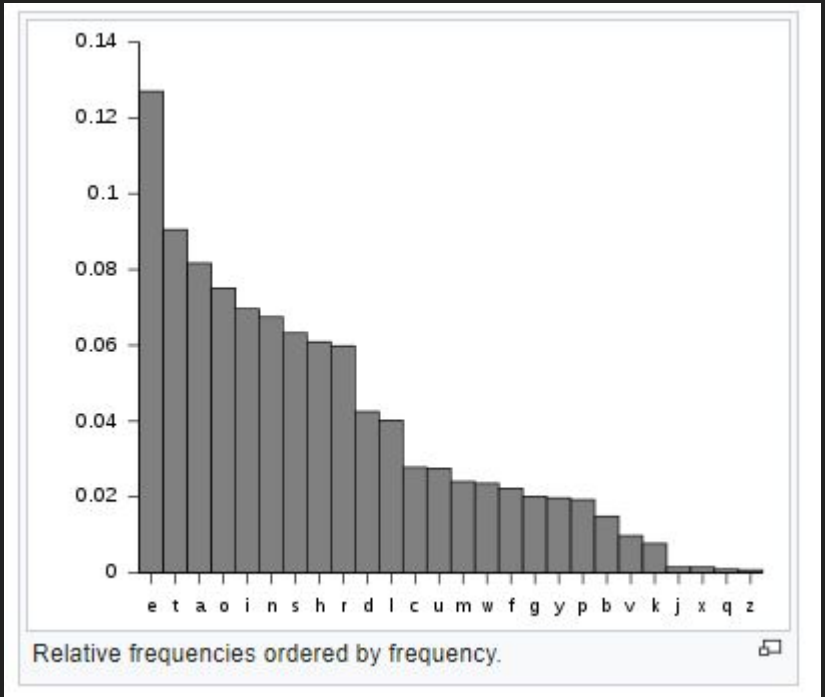
- Ciphertext

```
cei jvaql hkdtf udz yvoxr dsik cei npbw gdm
```

- Random map
- 26! possibility

Substitution cipher

1. Brute force when secret keys are not too many
2. Use letter frequency analysis when ciphertext is long



Substitution cipher

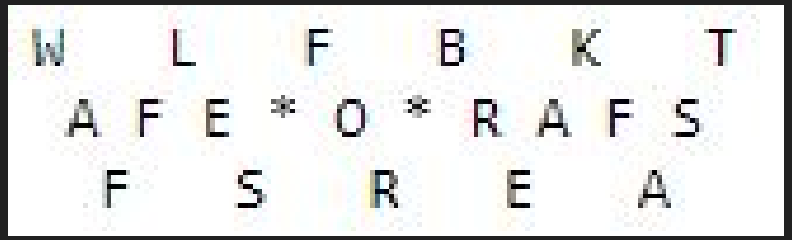
- Online tool
  - https://quipqiup.com/

Rail fence

- Plaintext

"WAFFLES FOR BREAKFAST"

- Ciphertext



Rail fence

- Online tool
  - http://rumkin.com/tools/cipher/railfence.php

JSFuck

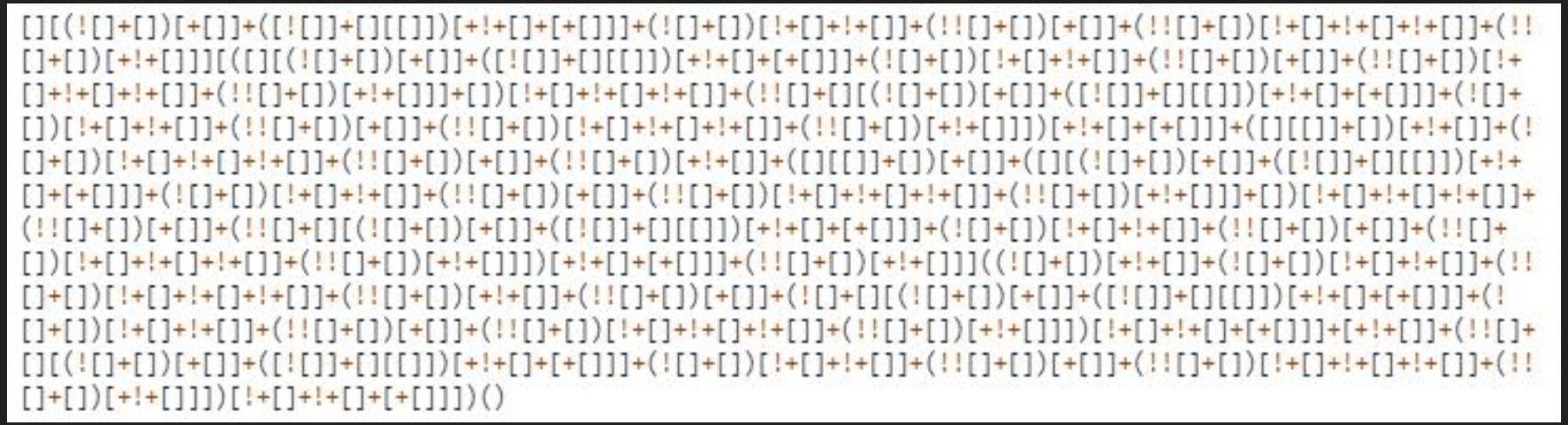
- Write javascript ONLY with [ ] ( ) ! +

Basics

- false => ![]
- true => !![]
- undefined => [][[]]
- NaN => +[![]]
- 0 => +[]
- 1 => +!+[]
- 2 => !+[]+!+[]
- 10 => [+!+[]]+[+[]]
- Array => []
- Number => +[]
- String => []+[]
- Boolean => ![]
- Function => []["filter"]
- eval => []["filter"]["constructor"](CODE)()
- window => []["filter"]["constructor"]("return this")()

JSFuck

- alert(1)



JSFuck

- Online tool
  - http://www.jsfuck.com/

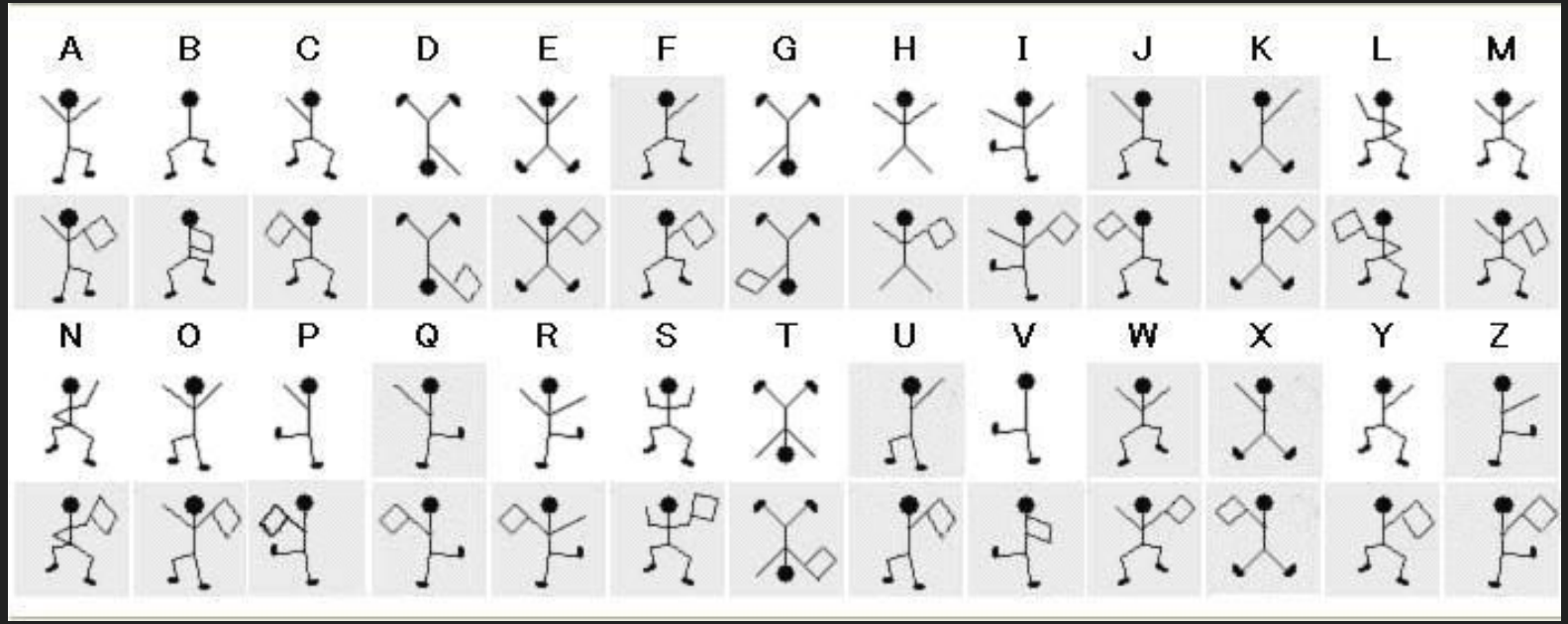
BrainFuck

brainfuck command	C equivalent
(Program Start)	`char array[INFINITELY_LARGE_SIZE] = {0};` `char *ptr=array;`
`>`	`++ptr;`
`<`	`--ptr;`
`+`	`++*ptr;`
`-`	`--*ptr;`
`.`	`putchar(*ptr);`
`,`	`*ptr=getchar();`
`[`	`while (*ptr) {`
`]`	`}`

BrainFuck

- Online tool
  - https://www.splitbrain.org/services/ook

Something interesting



eXclusive OR

XOR

- Plaintext ^ Pattern = Ciphertext
- Ciphertext ^ Pattern = Plaintext

XOR truth table

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

- 0, false
- 1, true

XOR

For example, the string "Wiki" (`01010111 01101001 01101011 01101001` in 8-bit ASCII) can be encrypted with the repeating key `11110011` as follows:

```
  01010111 01101001 01101011 01101001
⊕ 11110011 11110011 11110011 11110011
= 10100100 10011010 10011000 10011010
```

And conversely, for decryption:

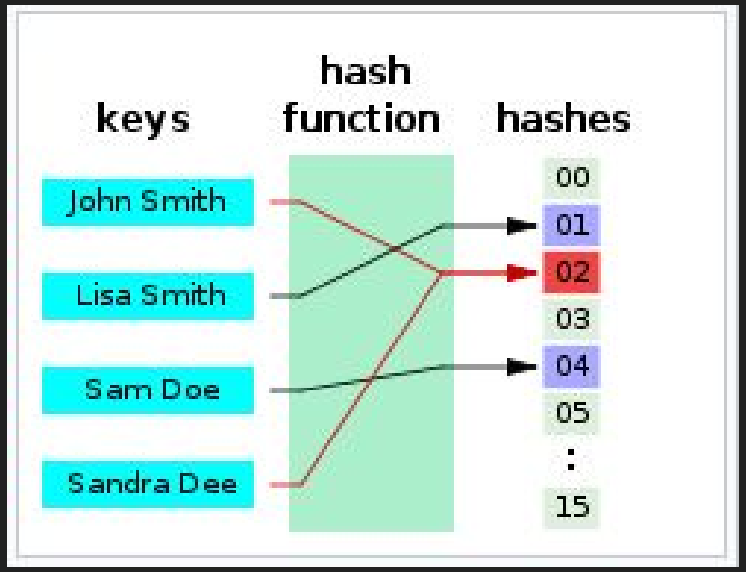
```
  10100100 10011010 10011000 10011010
⊕ 11110011 11110011 11110011 11110011
= 01010111 01101001 01101011 01101001
```

xortool

- Secret key is short than message
- The most frequent character
  - file 0x00
  - text 0x20
- https://github.com/hellman/xortool

Hash

Hash


keys
hash function
hashes
John Smith
Lisa Smith
Sam Doe
Sandra Dee
00
01
02
03
04
05
:
15

Hash collision

- SHA1
- https://shattered.it/

Rivest Shamir Adleman

RSA

1. Generate primes p, q and N = p * q
2. r = (p - 1) * (q - 1)
3. Choose e < r, gcd(e, r) = 1
4. e * d mod r = 1, find d

- Public key = (N, e)
- Private key = (N, d)

RSA

- Encrypt

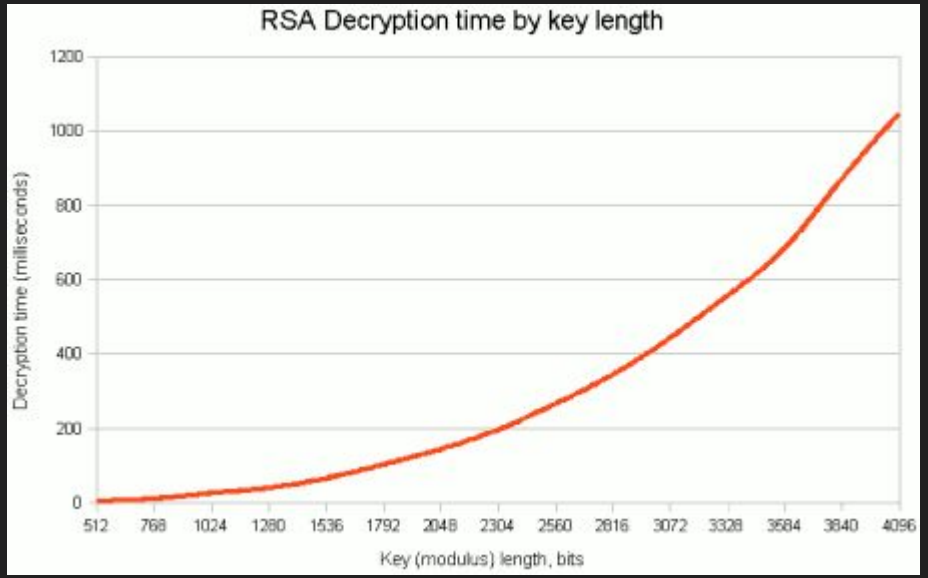
$$n^e \equiv c \pmod{N}$$

- Decrypt

$$c^d \equiv n \pmod{N}$$

Factor

- N is small
- Fact N to generate p and q
- https://factordb.com/



Other

Random seed

```
#!/usr/bin/python -u
import random,string

flag = "FLAG:"+open("flag", "r").read()[:-1]
encflag = ""
random.seed("random")
for c in flag:
  if c.islower():
    #rotate number around alphabet a random amount
    encflag += chr((ord(c)-ord('a')+random.randrange(0,26))%26 + ord('a'))
  elif c.isupper():
    encflag += chr((ord(c)-ord('A')+random.randrange(0,26))%26 + ord('A'))
  elif c.isdigit():
    encflag += chr((ord(c)-ord('0')+random.randrange(0,10))%10 + ord('0'))
  else:
    encflag += c
print "Unguessably Randomized Flag: "+encflag
```

Reference

- https://en.wikipedia.org/wiki/Base64
- https://en.wikipedia.org/wiki/Caesar_cipher
- https://en.wikipedia.org/wiki/Letter_frequency
- https://en.wikipedia.org/wiki/Brainfuck
- https://en.wikipedia.org/wiki/XOR_cipher
- https://en.wikipedia.org/wiki/Hash_function
- https://ctf-wiki.github.io/ctf-wiki/crypto/introduction.html
- https://2017game.picoctf.com

Q & A

Thanks for listening