Cryptography

A (nearly) complete overview

Gaspare Ferraro

March 31, 2019

Visit us!
zenhack.it

@GaspareG
ferraro@gaspa.re

Table of Contents

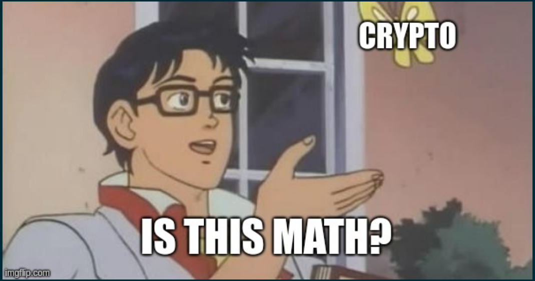
- 1. Introduction
- 2. Message encoding
- 3. Classical cryptography
- 4. Symmetric-key cryptography
- 5. Public-key cryptography
- 6. Key exchange
- 7. Hash function
- 8. Steganography

Table of Contents

- 1. Introduction
- 2. Message encoding
- 3. Classical cryptography
- 4. Symmetric-key cryptography
- 5. Public-key cryptography
- 6. Key exchange
- 7. Hash function
- 8. Steganography

Warning!

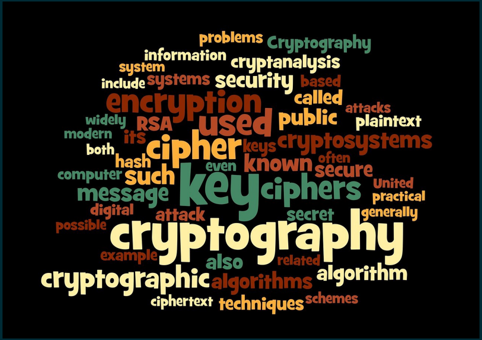
In this lesson we will use *maths*!



It wasn't always like that though ...

Why cryptography?

Cryptography (from greek: kryptos ”hidden, secret” and graphein, ”to write”)



The science of secure communication

Cryptography yesterday

(a) Cesare Chiper

(b) Scitala

Cryptography today

The needs, as well as the resources available, have evolved and today we can divide cryptography into:

(EN|DE)CRYPTION	**ASYMMETRIC (RSA, ECC, ...)** **SYMMETRIC (DES, AES, ...)**
KEY EXCHANGE	**RSA, DH, ECDH, ...**
AUTHENTICATION	**RSA, DSA, ECDSA, ...**
HASHING	**MD5, SHA-1, SHA-256, ...**

Table of Contents

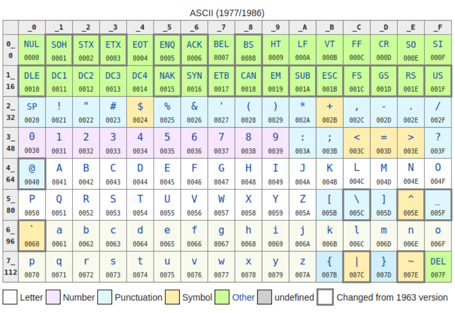
- 1. Introduction
- 2. Message encoding
- 3. Classical cryptography
- 4. Symmetric-key cryptography
- 5. Public-key cryptography
- 6. Key exchange
- 7. Hash function
- 8. Steganography

What is a message?

ASCII encoding

ASCII = American Standard Code for Information Interchange

char encoded in 7 bit + 1 bit for check (parity bit).



ASCII (1977/1986)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_ 0	NUL 0000	SOH 0001	STX 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
1_ 16	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
2_ 32	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3_ 48	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4_ 64	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5_ 80	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\\ 005C	] 005D	^ 005E	_ 005F
6_ 96	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7_ 112	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F

□ Letter □ Number □ Punctuation □ Symbol □ Other □ undefined □ Changed from 1963 version

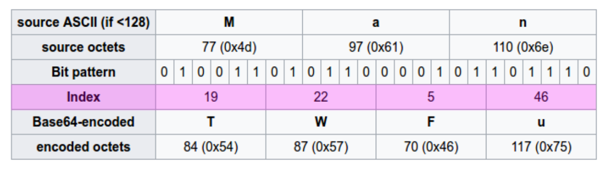
0, . . . , 31 + 127 → non-printable chars (null, new line, tab, others)
32, . . . , 126 → printable chars (letters, digits, punctuation, others)

Extended ASCII → char encoded in 8 bit (add 128 printable chars to standard ASCII)

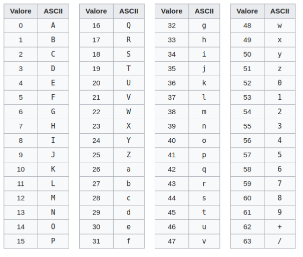
Unicode encoding

Base64

Advantage: encode all the ASCII chars in printable chars



<table>
<tr><th>source ASCII (if &lt;128)</th><th colspan="8">M</th><th colspan="8">a</th><th colspan="8">n</th></tr>
<tr><th>source octets</th><td colspan="8">77 (0x4d)</td><td colspan="8">97 (0x61)</td><td colspan="8">110 (0x6e)</td></tr>
<tr><th>Bit pattern</th><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><th>Index</th><td colspan="6">19</td><td colspan="6">22</td><td colspan="6">5</td><td colspan="6">46</td></tr>
<tr><th>Base64-encoded</th><td colspan="6">T</td><td colspan="6">W</td><td colspan="6">F</td><td colspan="6">u</td></tr>
<tr><th>encoded octets</th><td colspan="6">84 (0x54)</td><td colspan="6">87 (0x57)</td><td colspan="6">70 (0x46)</td><td colspan="6">117 (0x75)</td></tr>
</table>



Valore	ASCII	Valore	ASCII	Valore	ASCII	Valore	ASCII
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	/

Message are padded with =

flag → *ZmxhZwo=*

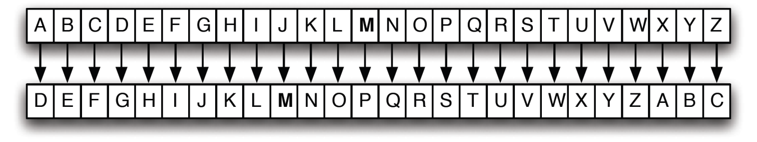
Table of Contents

- 1. Introduction
- 2. Message encoding
- 3. Classical cryptography
- 4. Symmetric-key cryptography
- 5. Public-key cryptography
- 6. Key exchange
- 7. Hash function
- 8. Steganography

Caesar cipher

Encrypt: left shift each letter of 3 positions

Decrypt: right shift each letter of 3 positions



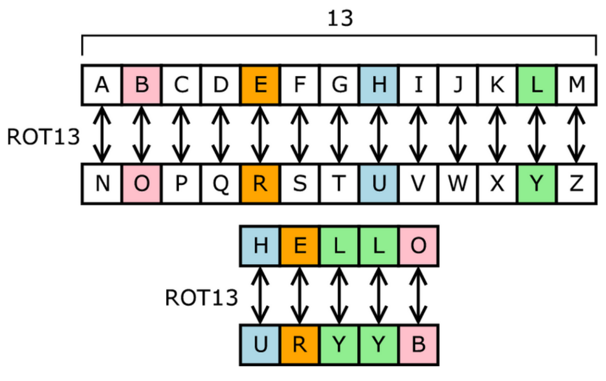
A	B	C	D	E	F	G	H	I	J	K	L	**M**	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	**M**	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

General cipher: shift letter of K positions

Attack: bruteforce all the possible K (only 26...)

ROT{13, 47}

ROT13: Caesar cipher with $K = 13$ on alphabetic dictionary
ROT47: Caesar cipher with $K = 47$ on printable ASCII chars (33 - 126).



Why $K = 13$ (or $K = 47$)? Because Encrypt = Decrypt

Substitution cipher

Monoalphabetic cipher: $C_{new} = P[C_{old}]$ (Where P is a dictionary permutation)

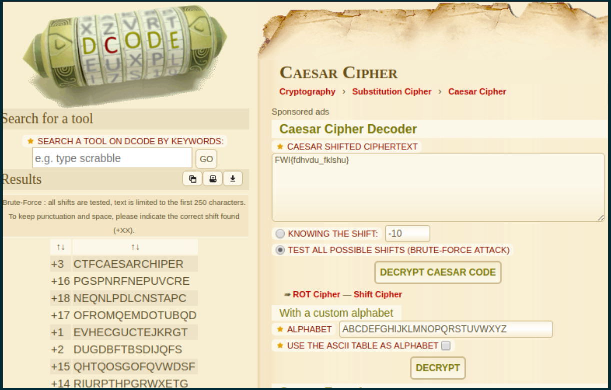
Polialphabetic cipher: multiple substitution alphabets (more than one dictionary permutation)

Cryptanalysis

https://quipqiup.com

dcode.fr

https://www.dcode.fr/tools-list



Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type scrabble GO

Results

Brute-Force : all shifts are tested, text is limited to the first 250 characters. To keep punctuation and space, please indicate the correct shift found (+XX).

↑↓	↑↓
+3	CTFCAESARCHIPER
+16	PGSPNRFNEPUVCRE
+18	NEQNLPDLCNSTAPC
+17	OFROMQEMDOTUBQD
+1	EVHECGUCTEJKRGT
+2	DUGDBFTBSDIJQFS
+15	QHTQOSGOFQVWDSF
+14	RIURPTHPGRWXETG

Caesar Cipher

Cryptography › Substitution Cipher › Caesar Cipher

Sponsored ads

Caesar Cipher Decoder

★ CAESAR SHIFTED CIPHERTEXT

FWI{fdhvdu_fklshu}

○ KNOWING THE SHIFT: -10

● TEST ALL POSSIBLE SHIFTS (BRUTE-FORCE ATTACK)

DECRYPT CAESAR CODE

➡ ROT Cipher — Shift Cipher

With a custom alphabet

★ ALPHABET ABCDEFGHIJKLMNOPQRSTUVWXYZ

★ USE THE ASCII TABLE AS ALPHABET □

DECRYPT

Almost all possible classic ciphers (old and new), encoder/decoder, ...

Table of Contents

- 1. Introduction
- 2. Message encoding
- 3. Classical cryptography
- 4. Symmetric-key cryptography
- 5. Public-key cryptography
- 6. Key exchange
- 7. Hash function
- 8. Steganography

Symmetric-key cryptography

Shannon principle

XOR cipher

One-time pad

Many-time pad

XorTool

Block vs Stream ciphers

DES

AES

Padding a message (PKCS#5 & PKCS#7)

How to handle messages of length not multiple of the block size?

Idea: append "some chars" to the message

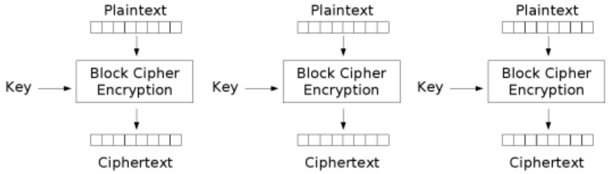
PKCS#5:
The padding string PS shall consist of $8 - (||M|| \mod 8)$ *octets all having value* $8 - (||M|| \mod 8)$.

PKCS#7:
For such algorithms, the method shall be to pad the input at the trailing end with $k - (l \mod k)$ *octets all having value* $k - (l \mod k)$*, where* l *is the length of the input.*

Why $8 - (||M|| \mod 8)$ and not $(||M|| \mod 8)$?

Block cipher mode of operation

ECB (Electronic Codebook)

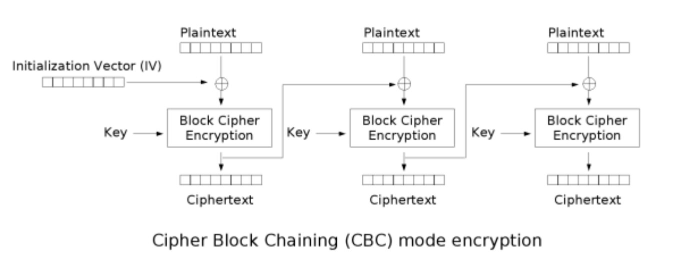


Electronic Codebook (ECB) mode encryption

$$C_i = f(M_i, Key)$$

How to break ECB (padding-oracle attack)

CBC (Cipher Block Chaining)



Cipher Block Chaining (CBC) mode encryption

$$C_i = f(M_i \oplus IV_i, Key)$$

$$IV_0 = \text{given in input (randomly generated)}$$

$$IV_{i+1} = C_i$$

How to break CBC (bit-flipping attack)