Ressource de Formation à la Cryptographie

Daresse (HERNANDEZ Thomas) pour club info INSA Toulouse

Année scolaire 2024-2025

Contents

1 Prérequis

Afin de pouvoir progresser confortablement dans cette formation, il est recommandé d'avoir des connaissances de base sur des concepts informatiques et de programmation tels que :

- algorithmique de base (boucles, variables, conditions)
- structures de données de base (listes/chaînes de caractères)
- python installé

Ces prérequis sont surtout importants pour la réalisation des exercices pratiques.

2 Introduction

Dans cette section, nous verrons ce qu'est, dans les grandes lignes, la cryptographie. Pour toute question ou volonté d'approfondissement n'hesitez pas à contacter vos respo sécu.

2.1 Définition

La cryptographie est la discipline qui consiste à rendre des données inutilisables pour toute personne autre que leur destinataire initial.

2.2 Exemple

L'algorithme de César : Si je vous donne la phrase suivante : "zljjbkq bpq slqob yixknrbqqb?", il y a peu de chance que vous sachiez quoi en faire. En revanche, si vous savez que j'utilise un algorithme de César de clef 23 (décalage dans l'alphabet de chaque lettre de 23 caractères), vous pourriez effectuer l'opération réciproque et obtenir "comment est votre blanquette?".

2.3 Préambule

Pour compléter ce parcours de formation il est recommandé d'y ajouter des exercices pratiques, pour ce faire le site web "cryptohack" offre une grande liste d'exercices et les chapitres de ce parcours de formation sont pensés pour suivre ceux de cryptohack. : https://cryptohack.org/.

2.4 lexique

- cryprographie : art de chiffrer une donnée.
- cryptanalyse : art de casser les systèmes de chiffrement. Ou accéder aux données claires à partir des données chiffrés sans l'algorithme ou la clef de déchiffrement.
- un message "clair" : message lisible par un être humain. ex : "je mange une pomme".
- un message "chiffré" : message protégé par un algorithme de chiffrement. ex: "mh pdqjh xqh srpph".
- chiffrement : processus par lequel on passe d'un message clair à un message chiffré.
- déchiffrement : processus inverse au chiffrement.
- encodage : processus de modification d'une donnée pour en faciliter son transport ou son utilisation (s'oppose au chiffrement dans le fait que ce n'est pas un moyen de protéger la donnée) ex: Base64.
- clef de chiffrement : donnée utilisée par l'algorithme de chiffrement nécessaire au chiffrement (ex: dans l'utilisation d'un code de césar la valeur du décalage des caractères dans l'alphabet est la clef de chiffrement ET de déchiffrement car elle permet également le déchiffrement ce qui n'est pas le cas dans les algorithmes de chiffrements dit "asymétriques")
- décryptage : processus de déchiffrement de force (sans connaitre la clef de chiffrement ou l'algorithme de chiffrement/déchiffrement)

- concaténer : mettre à la suite (ex: concaténation de "da" et de "do" : "dado")
- chiffrement asymétrique : type de chiffrement à clef dont la clef de chiffrement est différente de la clef de déchiffrement et ne permet pas le déchiffrement.
- chiffrement symétrique : type de chiffrement à clef unique où la clef de chiffrement est la même que la clef de déchiffrement.
- Attaque par clair-connu : Type d'attaque où l'attaquant possède une partie de la donnée chiffrée et déchiffrée.
- clair-connu : partie de la donnée claire que l'attaquant connait tout en connaissant également l'équivalent de cette donnée chiffré.

3 Chapitre 1 : Le Stockage Des Données Dans Un Ordinateur

Dans ce chapitre, nous allons essayer de comprendre ce qu'est la donnée du point de vue de l'ordinateur et donc du programme. Un ordinateur stocke les données du programme dans la RAM (Random Access Memory). Il n'est pas essentiel de comprendre comment celle-ci fonctionne pour ce parcours de formation.

Cependant, ce qu'il faut comprendre, c'est que pour l'ordinateur, toute donnée n'est qu'une suite de 0 et de 1 (base 2), ce qu'on appelle le système binaire. On peut alors se demander comment est-il possible de stocker du texte sous forme de nombre. Pour cela, l'ordinateur utilise une table de traduction comme la table ASCII présentée plus loin. Chaque caractère est codé sur 8 bits (Binary digIT : un 1 ou un 0), 8 bits formant ce que l'on appelle un octet (ex: 1011 0010). Ce qui offre $2^8 = 256$ possibilités.

On utilise rarement le système binaire en tant qu'humain car une suite de 0 et de 1 est particulièrement indigeste pour l'œil. C'est pour cela qu'on préfère le décimal, notre système habituel pour compter avec des chiffres de 0 à 9, ou bien l'hexadécimal ou base 16 dont les chiffres sont 0-9 et A-F, Cela permet de diviser par 4 la taille d'un "texte" binaire. En effet, chaque groupement de 4 bits correspond à un chiffre hexadécimal comme montré dans le tableau suivant.

Exemple:

Hexa	Bin	Dec	Hexa	Bin	Dec
00	0000 0000	0	1F	0001 1111	31
01	0000 0001	1	20	0010 0000	32
02	0000 0010	2	21	0010 0001	33
03	0000 0011	3	22	0010 0010	34
04	0000 0100	4	23	0010 0011	35
05	0000 0101	5	24	0010 0100	36
06	0000 0110	6	25	0010 0101	37
07	0000 0111	7	26	0010 0110	38
08	0000 1000	8	27	0010 0111	39
09	0000 1001	9	28	0010 1000	40
0A	0000 1010	10	29	0010 1001	41
0B	0000 1011	11	2A	0010 1010	42
0C	0000 1100	12	2B	0010 1011	43
0D	0000 1101	13	2C	0010 1100	44
0E	0000 1110	14	2D	0010 1101	45
0F	0000 1111	15	2E	0010 1110	46
10	0001 0000	16	2F	0010 1111	47
11	0001 0001	17	30	0011 0000	48
12	0001 0010	18	31	0011 0001	49
13	0001 0011	19	32	0011 0010	50
14	0001 0100	20	33	0011 0011	51
15	0001 0101	21	34	0011 0100	52
16	0001 0110	22	35	0011 0101	53
17	0001 0111	23	36	0011 0110	54
18	0001 1000	24	37	0011 0111	55
19	0001 1001	25	38	0011 1000	56
1A	0001 1010	26	39	0011 1001	57
1B	0001 1011	27	3A	0011 1010	58
1C	0001 1100	28	3B	0011 1011	59
1D	0001 1101	29	3C	0011 1100	60
1E	0001 1110	30	3D	0011 1101	61

3.1 La Table ASCII:

(la colonne "oct" correspond à l'écriture de la valeure numérique en base 8, celle-ci étant rarement utilisée)

Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char
0	00	000	0000000	NUL (null character)	32	20	040	0100000	space	64	40	100	1000000	@	96	60	140	1100000	`
1	01	001	0000001	SOH (start of header)	33	21	041	0100001	!	65	41	101	1000001	A	97	61	141	1100001	a
2	02	002	0000010	STX (start of text)	34	22	042	0100010	"	66	42	102	1000010	B	98	62	142	1100010	b
3	03	003	0000011	ETX (end of text)	35	23	043	0100011	#	67	43	103	1000011	C	99	63	143	1100011	c
4	04	004	0000100	EOT (end of transmission)	36	24	044	0100100	$	68	44	104	1000100	D	100	64	144	1100100	d
5	05	005	0000101	ENQ (enquiry)	37	25	045	0100101	%	69	45	105	1000101	E	101	65	145	1100101	e
6	06	006	0000110	ACK (acknowledge)	38	26	046	0100110	&	70	46	106	1000110	F	102	66	146	1100110	f
7	07	007	0000111	BEL (bell (ring))	39	27	047	0100111	'	71	47	107	1000111	G	103	67	147	1100111	g
8	08	010	0001000	BS (backspace)	40	28	050	0101000	(	72	48	110	1001000	H	104	68	150	1101000	h
9	09	011	0001001	HT (horizontal tab)	41	29	051	0101001	)	73	49	111	1001001	I	105	69	151	1101001	i
10	0A	012	0001010	LF (line feed)	42	2A	052	0101010	*	74	4A	112	1001010	J	106	6A	152	1101010	j
11	0B	013	0001011	VT (vertical tab)	43	2B	053	0101011	+	75	4B	113	1001011	K	107	6B	153	1101011	k
12	0C	014	0001100	FF (form feed)	44	2C	054	0101100	,	76	4C	114	1001100	L	108	6C	154	1101100	l
13	0D	015	0001101	CR (carriage return)	45	2D	055	0101101	-	77	4D	115	1001101	M	109	6D	155	1101101	m
14	0E	016	0001110	SO (shift out)	46	2E	056	0101110	.	78	4E	116	1001110	N	110	6E	156	1101110	n
15	0F	017	0001111	SI (shift in)	47	2F	057	0101111	/	79	4F	117	1001111	O	111	6F	157	1101111	o
16	10	020	0010000	DLE (data link escape)	48	30	060	0110000	0	80	50	120	1010000	P	112	70	160	1110000	p
17	11	021	0010001	DC1 (device control 1)	49	31	061	0110001	1	81	51	121	1010001	Q	113	71	161	1110001	q
18	12	022	0010010	DC2 (device control 2)	50	32	062	0110010	2	82	52	122	1010010	R	114	72	162	1110010	r
19	13	023	0010011	DC3 (device control 3)	51	33	063	0110011	3	83	53	123	1010011	S	115	73	163	1110011	s
20	14	024	0010100	DC4 (device control 4)	52	34	064	0110100	4	84	54	124	1010100	T	116	74	164	1110100	t
21	15	025	0010101	NAK (negative acknowledge)	53	35	065	0110101	5	85	55	125	1010101	U	117	75	165	1110101	u
22	16	026	0010110	SYN (synchronize)	54	36	066	0110110	6	86	56	126	1010110	V	118	76	166	1110110	v
23	17	027	0010111	ETB (end transmission block)	55	37	067	0110111	7	87	57	127	1010111	W	119	77	167	1110111	w
24	18	030	0011000	CAN (cancel)	56	38	070	0111000	8	88	58	130	1011000	X	120	78	170	1111000	x
25	19	031	0011001	EM (end of medium)	57	39	071	0111001	9	89	59	131	1011001	Y	121	79	171	1111001	y
26	1A	032	0011010	SUB (substitute)	58	3A	072	0111010	:	90	5A	132	1011010	Z	122	7A	172	1111010	z
27	1B	033	0011011	ESC (escape)	59	3B	073	0111011	;	91	5B	133	1011011	[	123	7B	173	1111011	{
28	1C	034	0011100	FS (file separator)	60	3C	074	0111100	<	92	5C	134	1011100	\	124	7C	174	1111100	|
29	1D	035	0011101	GS (group separator)	61	3D	075	0111101	=	93	5D	135	1011101	]	125	7D	175	1111101	}
30	1E	036	0011110	RS (record separator)	62	3E	076	0111110	>	94	5E	136	1011110	^	126	7E	176	1111110	~
31	1F	037	0011111	US (unit separator)	63	3F	077	0111111	?	95	5F	137	1011111	_	127	7F	177	1111111	DEL

Figure 1: Table ASCII

4 Chapitre 2 : l'encodage d'un message en base64

Dans ce chapitre nous allons voir en détail l'encodage et desencodage d'un message en base64 afin que cela devienne moins abstrait.

La base64 est une manière d'encore plus compacter les données que l'hexa utilisant une base constituée de 64 caractères ASCII.

Partons d'un message simple: "je mange une pomme", afin d'encoder ce message l'ordinateur va commencer par le traduire dans un language qu'il comprend : le binaire

caractère	decimal	binaire
j	106	0110 0101
e	101	0110 0101
	32	0010 0000
m	109	0110 1101
a	97	0110 0001
n	110	0110 1110
g	103	0110 0011
e	101	0110 0101
	32	0010 0000
u	117	0111 0101
n	110	0110 0001
e	101	0110 0101
	32	0010 0000
P	112	0111 0000
o	111	0111 0001
m	109	0110 1101
m	110	0110 1101
e	101	0110 0101

Ensuite, l'ordinateur va concaténer les binaires pour créer une grande chaine de 0 et de 1 : 011010100110010100100000011011010110000101101110011001110110010100
1000000111010101101110011001010010000001110000011011110110110101101101011001 01
(On comprend alors vite l'intéret d'utiliser l'hexa pour économiser de la place.)

O va ensuite séparer la chaine en groupes de 6 bits ce qui offre $2^6 = 64$ possibilités par groupements d'où l'utilisation de la Base64. On obtient donc :
011010 100110 010100 100000 011011 010110 000101 101110 011001 110110
010100 100000 011101 010110 111001 100101 001000 000111 000001 101111
011011 010110 110101 100101

On écrit ensuite tous ces nombres en décimal:
26 38 20 32 27 22 5 46 25 54 20 32 29 22 57 37 8 7 1 47 27 22 53 37
puis on écrit pour chacun leur correspondant dans la base64 ce qui nous donne:
a m U g b W F u Z 2 U g d W 5 l I H B v b W 1 l C g o
La chaine de caractère correspondant à "je mange une pomme" en Base64 est :
"amUgbWFuZ2UgdW5lIHBvbW1lCgo="

exercice : entrainez vous à faire l'opération inverse et vérifiez bien que vous retombez sur le bon message.

Sextet (déc.)	Code	Sextet (déc.)	Code	Sextet (déc.)	Code	Sextet (déc.)	Code
000000 (0)	A	000001 (1)	B	000010 (2)	C	000011 (3)	D
000100 (4)	E	000101 (5)	F	000110 (6)	G	000111 (7)	H
001000 (8)	I	001001 (9)	J	001010 (10)	K	001011 (11)	L
001100 (12)	M	001101 (13)	N	001110 (14)	O	001111 (15)	P
010000 (16)	Q	010001 (17)	R	010010 (18)	S	010011 (19)	T
010100 (20)	U	010101 (21)	V	010110 (22)	W	010111 (23)	X
011000 (24)	Y	011001 (25)	Z	011010 (26)	a	011011 (27)	b
011100 (28)	c	011101 (29)	d	011110 (30)	e	011111 (31)	f
100000 (32)	g	100001 (33)	h	100010 (34)	i	100011 (35)	j
100100 (36)	k	100101 (37)	l	100110 (38)	m	100111 (39)	n
101000 (40)	o	101001 (41)	p	101010 (42)	q	101011 (43)	r
101100 (44)	s	101101 (45)	t	101110 (46)	u	101111 (47)	v
110000 (48)	w	110001 (49)	x	110010 (50)	y	110011 (51)	z
110100 (52)	0	110101 (53)	1	110110 (54)	2	110111 (55)	3
111000 (56)	4	111001 (57)	5	111010 (58)	6	111011 (59)	7
111100 (60)	8	111101 (61)	9	111110 (62)	+	111111 (63)	/

Figure 2: table de conversion binaire/decimal -> base64, (NB : Sextet = groupement de 6bits)

4.1 à retenir

- prendre l'habitude de bien comprendre chaque étape d'un processus permet d'avoir une vision claire de ce que l'on fait ce qui est primordial afin de ne pas stagner sur des problèmes facilement résolubles.
- Il est important de bien faire la différence entre le message ("je mange une pomme") et la donnée sous toute les formes qu'elle peut prendre, une donnée pouvant donner plusieurs messages différent dépendamment de la manière dont elle sera traduite (base64 table ascii UTF-8 etc)

5 chapitre 3 : introduction sur les opérateurs bit à bit (bitwise operators)

Les opérateurs bit à bit sont des outils utilisés dans la programmation et l'informatique pour manipuler directement les bits des données binaires. Ces opérateurs travaillent au niveau des bits un par un. afin de rendre cela plus clair commençons avec des exemples:

- AND : a pour résultat 1 si les deux bits sont à 1 sinon renvoie 0 (symb : &):

bit 1	bit 2	AND
0	0	0
0	1	0
1	0	0

- OR : a pour résultat 1 si au moins un des bits vaut 1 et 0 sinon (symb : |):

bit 1	bit 2	OR
0	0	0
0	1	1
1	0	1
1	1	1

- XOR (ou exclusif): a pour résultat 1 si les deux bits sont différent et 0 sinon (symb : ^):

bit 1	bit 2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Exemple sur des octets :

octet 1	octet 2	operateur	resultat
1010 1011	1001 0110	AND	1000 0010
1010 1011	1001 0110	OR	1011 1111
1010 1011	1001 0110	XOR	0011 1101

Chaque opération se fait indépendamment sur chaque bit, commençant par effectuer l'opération sur les deux premiers puis les deux seconds etc. Il est également possible, en python par exemple d'utiliser ces opérateurs directement sur des Integers exemple 5=0101 13=1101 et 5^13 = 1000 = 8

Pour les amateurs de mathématiques tout cela se réfère à l'algèbre booléenne et à la logique mathématiques que nous n'approfondirons pas ici.

6 Chapitre 4 : L'opérateur XOR : des propriétés intéressantes

L'opérateur XOR est souvent utilisé en cryptographie notamment dans un des exemples les plus simples de chiffrement symétriques : le "chiffrement XOR" consistant à avoir une clef codée sur n bits (n variable) par exemple 1010 et à "XORer" notre donnée avec cette clef. Ainsi si notre donnée est 0110 1011 on obtient 1010^0110 = 1100 ; 1010^1011 = 0001 notre donnée chiffrée est donc 1100 0001. maintenant on peut se demander comment déchiffrer cette donnée. Une des propriétés qui rend l'opérateur XOR si interessant est que si on prend une donnée et qu'on la XOR deux fois avec la même clef on retombe sur la donnée initiale. En effet reprennons notre exemple précédent :

On a 1100 0001 et on XOR avec 1010

1100^1010 = 0110

0001^1010 = 1011

On retombe bien sur 0110 1011 qui était notre donnée initiale.

Il est vivement recommandé de se renseigner sur les propriétés mathématiques des opérateurs notamment pour la cryptanalyse, celles-ci pouvant être à l'origine de faille dans les algorithmes de chiffrements. Les pages Wikipédia respectives de ces opérateurs résument parfaitement les propriétés mathématiques de ceux-ci. (associativité, commutativité etc.)

7 chapitre 5 : Premier contact avec la cryptanalyse: L'attaque par clair-connu ou KPA

Afin de commencer simplement reprennons l'exemple de l'introduction avec le code de César. Si je vous donne le texte suivant : zljjbkq bpq slqob yixknrbqqb? mais que je vous dis que la première lettre du message clair est 'c' (ce 'c' est votre clair connu), vous savez que l'algorithme fait la correspondance :

$c \rightarrow z$

ce qui correspond à un décalage de 23 caractères dans l'alphabet ainsi en effectuant le décalage inverse sur le reste du message chiffré (remonter de 23 caractères dans l'alphabet) on arrive à trouver un message cohérent qui est : "comment est votre blanquette?" que l'on devine être le clair.

le chiffrement XOR est également assez vulnérable aux attaques par clair connu étant donnée que si on a un clair A et un "chiffré" B (= A^CLEF) on peut simplement effectuer l'opération A^B ce qui revient à faire A^A^CLEF pour obtenir la CLEF. (CF : propriétés de l'opérateur XOR)

7.1 conclusion

- Pour pouvoir effectuer une attaque par clair connu il faut que la clef ou l'algorithme que l'on cherche à déduire du couple (clair/chiffré) soit contenu intégralement au sein de ce couple, si on avait eu comme clef un ensemble de nombre ex: 23,12,8,24,2 où la première lettre serait décalée de 23 caractères la Deuxième de 12 caractères la troisième de 8 etc on aurait pas pu déduire cette clef de la 1ere lettre uniquement.
- Il est important de ne pas créer de paternes dans ces données afin d'éviter d'offrir un clair connu aux potentiels assaillants qui essaieraient de s'en prendre à votre système de sécurité
- anecdote : Pendant la 2nde guerre mondiale l'attaque par clair connu a été d'une grande aide dans le cassage du code enigma utilisé par les nazis, leur rapport météo matinal se terminant tous de la même manière ce qui a offert un clair connu aux alliés.