

Département d'Informatique

Informatique 2 SPC

2016-2017

TD3 – Fichiers et cryptographie

Dans cette série d'exercices il s'agira d'essayer plusieurs techniques de chiffrements alphabétiques d'un fichier texte initial en clair.

1. Le chiffrement de César

C'est la technique de chiffrement que vous utilisiez à l'école primaire pour échanger vos messages secrètement avec vos camarades.



Ce chiffrement consiste tout simplement à substituer chaque lettre dans l'alphabet par celle se trouvant à x crans d'elle (dans l'ordre alphabétique), où x est justement la clé secrète de « chiffrement » (avec $1 \leq x \leq 25$). Pour les dernières lettres, on reprend au début.

Avec cette technique de chiffrement, seules les lettres d'un texte sont remplacées (en prenant en compte la casse) ; tous les autres symboles sont inchangés.

Dans la suite nous prendrons pour convention de nommer nos fichiers avec les extensions suivantes :

- .txt quand il s'agira de texte clair (non chiffré)
- .ces quand il s'agira de texte chiffré avec la méthode de César

- 1) Définir une fonction **chiffreCaract** qui prend en entrée un caractère et un entier constituant la clé et qui retourne le caractère chiffré avec cette clé si cela est possible. Un symbole reste inchangé. Une majuscule est substituée par une majuscule et une minuscule par une minuscule.

Une bonne solution consiste à s'appuyer sur le codage ASCII des caractères.

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	\$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-`	63	3F	?	95	5F	_	127	7F	DEL

- 2) Écrire la fonction **saisieFichierExistant** qui prend en paramètre un message et demande à l'utilisateur de saisir au clavier un nom de fichier en conformité avec ce message. **Si le fichier existe** la fonction retourne une chaîne contenant le nom du fichier à lire et sinon elle retourne la valeur `null`.
- 3) Écrire la fonction **saisieFichierNonExistant** qui prend en paramètre un message et demande à l'utilisateur de saisir au clavier un nom de fichier en conformité avec ce message. **Si le fichier n'existe pas** la fonction retourne une chaîne contenant le nom du fichier à créer et sinon elle retourne la valeur `null`.
- 4) Écrire la fonction **chiffreFichier** qui demande à l'utilisateur le nom du fichier clair et le nom de fichier à chiffrer. Elle retourne la valeur `false` si le fichier clair n'existe pas ou si le fichier à chiffrer existe déjà. Cette fonction demande aussi à l'utilisateur de choisir la clé commune de chiffrement/déchiffrement (c'est-à-dire le décalage entre 1 et 25). Elle utilise ensuite la fonction `chiffreCaract` pour réaliser le chiffrement du fichier. Si tout se passe bien cette fonction retourne `true`.
- 5) Écrire la fonction **dechiffreFichier** qui demande à l'utilisateur le nom du fichier chiffré et le nom de fichier à déchiffrer. Elle retourne la valeur `false` si le fichier chiffré n'existe pas ou si le fichier à déchiffrer existe déjà. Cette fonction demande aussi à l'utilisateur de choisir la clé commune de chiffrement/déchiffrement (c'est-à-dire le décalage entre 1 et 25). Elle utilise ensuite la fonction `chiffreCaract` pour réaliser le déchiffrement du fichier. On notera en effet qu'un déchiffrement de x est tout simplement l'inverse d'un chiffrement de $26-x$ et réciproquement. Si tout se passe bien cette fonction retourne `true`.
- 6) Écrire la fonction **casserChiffrement** qui demande à l'utilisateur le nom du fichier chiffré dont on veut casser la clé et qui teste toutes les clés possibles et affiche les 3 premières lignes à l'écran pour chaque clé testée. Il appartiendra à l'utilisateur de retrouver la clé (pas très difficile puisqu'un seul message devrait avoir du sens). Une fois la clé trouvée, l'utilisateur pourra utiliser la fonction de déchiffrement pour avoir la totalité du fichier. La fonction de cette question retourne la valeur `false` si le fichier chiffré n'existe pas. Elle retourne `true` sinon.
- 7) Écrire le programme principal qui proposera un menu à l'utilisateur, en faisant appel aux fonctions précédentes, pour lui permettre de faire différentes opérations cryptographiques sur les fichiers qu'il choisira. On pensera à traiter les erreurs des appels aux différentes fonctions

Le menu proposera :

- De chiffrer un fichier
- De déchiffrer un fichier
- De casser le chiffrement d'un fichier
- De quitter le programme

Question supplémentaire : Changer le programme pour que la clé de chiffrement/déchiffrement puisse évoluer depuis une clé initiale à chaque caractère chiffré/déchiffré. Par exemple pour faire avancer la clé de 5 à chaque caractère chiffré/déchiffré. Ceci peut rendre plus difficile une analyse sur la fréquence d'apparition des caractères (même si dans le cas présent, le nombre de clés à tester en force brute sera toujours le même).