Enigma

bit.ly/Projet_Enigma

Christopher Marshall-Breton
christopher.marshallbreton@gmail.com

Enigma est une machine de chiffrement et de déchiffrement d'origine allemande. Basée sur un système électromécanique et utilisant des rotors, elle a permis de chiffrer les communications allemandes pendant la seconde guerre mondiale, et a été à l'origine des premières "bombes" électromécaniques et de l'utilisation de machines pour leur puissance de calcul, sous la direction d'un certain Alan Turing.

Dans ce projet, nous allons simuler une version extrêmement simplifiée d'Enigma.

Dans un premier temps, vous allez jouer le rôle des Allemands, et créer des algorithmes pour chiffrer et déchiffrer des messages grâce à Enigma.

Dans un deuxième temps, vous allez jouer le rôle des Anglais, et essayer de cracker le code, sans connaître les configurations d'Enigma.

Enigma :

1. Fonctionnement

La version historique d'Enigma détaillée sur plusieurs sites (Wikipédia, AeL, bibmath…)

Dans notre cas, nous allons restreindre Enigma à **2 rotors**.
Chaque rotor contient toutes les lettres de l'alphabet dans le désordre. Ils nous servent à coder et décoder les messages.
ATTENTION ! Les rotors font partie intégrante de la machine ! Si vous perdez ou modifiez vos rotors, le message ne peut plus être déchiffré !!

Pour coder un message, il faut prendre chaque lettre séparément et la comparer au premier rotor, puis au deuxième.
Pour décoder le message, on fait le processus inverse.

Le détail du mécanisme est présenté en annexe.

NOTE : La force d'enigma est que, même avec la machine, on ne peut pas décoder un message si l'on ne connaît pas la configuration initiale. pour enigma_code() et enigma_decode(), il faudra donc passer en paramètre **les rotors ET leur configuration initiale** !

2. Rotors

Un Rotor est un tableau contenant l'alphabet complet dans un ordre mélangé.
On commence par créer 2 rotors, ces deux rotors seront utilisés INCHANGÉS pour le reste des algos.

On débute donc avec 2 tableaux contenant l'alphabet dans un ordre mélangé (chaque rotor est différent).

Pour la version enigma de code/decode, il faudra connaître la configuration des rotors. Cela veut dire comment les rotors sont positionnés au moment où l'on commence à coder/décoder.
Exemple :

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

Rotor 1

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

Rotor 2

On commence à coder dans cette configuration (Rotor 1 : A, Rotor 2 : P).

Après avoir codé, on se retrouve avec les rotors comme suit :

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

Rotor 1

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

Rotor 2

Les rotors sont donc en configuration (Rotor 1 : S, Rotor 2 : L)
On ne peut pas décoder le message si l'on ne se replace pas en configuration de départ (Rotor 1 : A, Rotor 2 : P). Il faut donc toujours réinitialiser les rotors dans la position utilisée pour coder le message. => Il faut passer cette configuration en paramètre.

Partie 1 : Le code

Objectifs :

On veut :

- Un algorithme ***shuffle_alphabet*** qui mélange l'alphabet dans un tableau pour créer un rotor.
- Un algorithme ***code*** qui code un message en le faisant passer par 2 rotors SANS mouvement
- Un algorithme ***decode*** qui décode le message en utilisant les 2 mêmes rotors SANS mouvement
- Un algorithme ***enigma_code*** qui code un message en le faisant passer par 2 rotors qui tournent, AVEC leur configuration de départ
- Un algorithme ***enigma_decode*** qui décode le message en utilisant les 2 mêmes rotors qui tournent, AVEC leur configuration de départ
- Un algorithme ***turing_decode*** qui déduit le message avec rotors mais SANS la configuration, en utilisant un mot probable.

ATTENTION ! Le code doit être testable sur console ou par Thonny sans interface graphique. Ne commencez pas la partie 2 si la partie 1 n'est pas finie.

Principe des algos

Step 1 : Enigma statique

L'algorithme **code**() prend en paramètre un message et les deux rotors. Il code ensuite le message par le principe suivant :

1. Je regarde la lettre à coder. Je cherche à quelle lettre elle correspond dans le rotor 1 (pour C, je regarde la 3e lettre du rotor)
2. Je regarde la nouvelle lettre à coder. Je cherche à quelle lettre elle correspond dans le rotor 2 (pour C, je regarde la 3e lettre du rotor)

L'algorithme **decode**() prend en paramètre un message et les deux rotors Il code ensuite le message par le principe suivant :

1. Je regarde la lettre à décoder. Je cherche à quelle place elle correspond dans le rotor 2 et j'en déduis à quelle lettre ça correspond (si c'est la 3e lettre du rotor, je transforme en C)
2. Je regarde la lettre à décoder. Je cherche à quelle place elle correspond dans le rotor 1 et j'en déduis à quelle lettre ça correspond (si c'est la 3e lettre du rotor, je transforme en C)

Finalement, on n'a pour l'instant qu'une simple permutation. On a ici un chiffre de César.
Ce qui veut dire que cracker ce code est extrêmement facile, surtout pour des messages longs.

Step 2 : Enigma en mouvement

Les algorithmes **enigma_code**() et **enigma_decode**() se comportent respectivement comme code() et decode(), à une exception près: **les rotors tournent** !
A chaque lettre codée, le rotor 1 tourne d'un cran. Lorsqu'il a effectué un tour complet, le rotor 2 tourne d'un cran, etc (un peu comme les secondes et les minutes sur une horloge).
Cela signifie que la même lettre peut être codée en deux lettres différentes.

ATTENTION : pour décoder, on passe par le même principe, ce qui implique de connaître les configurations d'origine des rotors. enigma_code() et enigma_decode() doivent donc prendre en paramètre la configuration initiale des rotors (leur première lettre par exemple)

Lorsque l'on décode, pour chaque lettre, on avance le rotor 1 d'un cran, puis le 2 d'un cran lorsque le premier a fait un tour complet.

Step 3 : Cracker le code

Pour cracker nos messages, on va s'aider un peu et faire une hypothèse :

- On considère que l'on a à disposition un mot du message.
    - Exemple : `RAPPORT DE PATROUILLE DU SOUS MARIN NAUTILUS: PREMIER CONTACT AU NORD DE BREST. MER CALME. CHALUTIER ISOLE. PAS D ENGAGEMENT. DEUXIEME CONTACT AU SUD DE BRIGHTON. CONVOI MARCHAND. TROIS NAVIRES COULES. ESCORTE EN DEROUTE. INTEMPERIES SUR LE RETOUR. AVARIE EXTERNE TUBE TORPILLE SUPERIEUR GAUCHE. BALLAST ENDOMMAGE. RETOUR A LORIENT LUNDI MIDI. PROCHAINE PATROUILLE A L EST DE HORNSEA TRANSMISSION TERMINEE`
- On part du principe que l'on SAIT que notre message contient le mot "TORPILLE"

L'algorithme **turing_decode**() prend en paramètre un message codé , les rotors, le message probable, mais PAS les configurations de départ des rotors. L'algorithme va donc essayer toutes les positions de départ des rotors jusqu'à trouver le message en clair. L'algo saura que le message est décodé quand il contiendra le message probable.

Il faut également que l'algorithme donne les configurations initiales des rotors. Un retour multiple est possible. (en Python)

Partie 2 : Le visuel

Objectifs de la partie graphique :

On veut :

- Un programme pour coder.
- Un programme pour décoder.

Les deux utilisent le fichier Enigma.py contenant les codes de la partie 1.

Il NE FAUT QUE des problématiques graphiques dans ces programmes. Tous les problèmes liés au fonctionnement d'Enigma sont dans Enigma.py.

Le détail de l'interface est laissé libre. Seul le fonctionnel est demandé.

Les fonctionnalités demandées sont les suivantes :

- Une zone où écrire le message à coder/décoder.
- Une zone sélectionnable où afficher le message de sortie (on veut pouvoir copier-coller le message en sortie).
- Un bouton permettant de coder le message SANS mouvement des rotors.
- Un bouton permettant de coder le message AVEC mouvement des rotors.
- Un affichage des Rotors.
- Permettre de choisir la configuration des rotors.
- Un bouton permettant de décoder le message sans les configurations initiales par un champ où saisir un mot probable.
- Le fonctionnement lettre à lettre.

Pour aller (beaucoup) plus loin :

En réalité, Enigma possède 5 rotors, dont 3 fonctionnels, ce qui veut dire que pour décoder, il faut non seulement connaître les positions initiales des rotors, mais savoir QUELS rotors on utilise ! (3 sur 5)

Cela signifie que l'on va devoir tester toutes les configurations de départ, ET toutes les configurations de rotors !

L'objectif maintenant est de modifier nos algos précédents pour avoir 5 rotors dont 2 fonctionnels. On va donc passer les configurations initiales, et le numéro des rotors utilisés.
L'algo turing_decode() va donc devoir chercher les configs initiales start + rotors !

Mise en forme :

Les messages étaient codés par blocs de 4 lettres, modifiez vos algos pour pouvoir chiffrer/déchiffrer un message formaté.

Et pour les vrais cracks :

Autant rajouter les câbles de permutation et le réflecteur !
ATTENTION : le cassage de code va évidemment changer.

Conseils pour l'ensemble du projet

- On ne s'intéresse qu'à des messages en majuscules. Si vous rencontrez des caractères spéciaux, ne les prenez pas en compte, ou transformez les en espace pour plus de lisibilité.
- Attention aux paramètres et aux retours de vos algos !!
- Si vous rencontrez un caractère spécial (espace, accents, point, apostrophe…) transformez le en espace (sans rotation de rotor)
- Essayez sur papier si vous doutez.
- Pour tester vos programmes, essayez de coder un code avec le programme d'un groupe et de le faire décoder par vos camarades.

Différences avec la "VRAIE" machine Enigma :

- Une lettre ne peut jamais être codée en elle même : A ne donnera jamais A
- Il y a 3 rotors parmi 5
- Des câbles rajoutent des permutations arbitraires (A devient T, B devient N, C devient G etc…)
- Les lettres font un aller-retour par les rotors (réflecteur)

ANNEXES

Exemple de code/décode

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

Alphabet

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

Rotor 1

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

Rotor 2

Coder le message : "BONJOUR"

- On regarde la première lettre :B
- B est en 2è position dans l'alphabet, on regarde la lettre en 2è position dans le Rotor_1 : B se transforme donc en Q
- Q est en 17è position dans l'alphabet, on regarde la lettre en 17è position dans le Rotor_2 : Q se transforme en F
- La première lettre est donc passée de B à F

- On regarde la deuxième lettre :O
- O est en 15è position dans l'alphabet, on regarde la lettre en 15è position dans le Rotor_1 : O se transforme donc en B
- B est en 2è position dans l'alphabet, on regarde la lettre en 2è position dans le Rotor_2 : B se transforme donc en O
- La deuxième lettre est donc passée de O à O (oui, ça peut arriver dans notre cas)

On continue pour le reste des lettres : BONJOUR devient FORDOZJ

On veut décoder "FORDOZJ" en utilisant les mêmes rotors !

- On regarde la première lettre :F
- F est en 17è position dans le Rotor_2, on regarde la lettre en 17è position dans l'alphabet : F se transforme donc en Q
- Q est en 2è position dans le Rotor_1, on regarde la lettre en 2è position dans l'alphabet : Q se transforme en B
- La première lettre est donc passée de F à B

- On regarde la deuxième lettre :O
- O est en 2è position dans le Rotor_2, on regarde la lettre en 2è position dans l'alphabet : O se transforme donc en B

- B es en 15è position dans le Rotor_1, on regarde la lettre en 15è position dans l'alphabet : B se transforme en O
- La deuxième lettre est donc passée de O à O

On continue pour le reste des lettres : FORDOZJ devient BONJOUR

Exemple de code AVEC MOUVEMENT

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

Alphabet

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

Rotor 1

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

Rotor 2

Coder le message : "BONJOUR" avec **positions initiales R1 : 'A' et R2 : 'P'**

- On regarde la première lettre :B
- B est en 2è position dans l'alphabet, on regarde la lettre en 2è position dans le Rotor_1 : B se transforme donc en Q
- Q es en 17è position dans l'alphabet, on regarde la lettre en 17è position dans le Rotor_2 : Q se transforme en F
- La première lettre est donc passée de B à F

LE ROTOR 1 TOURNE D'UN CRAN !

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

Rotor 1

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

Rotor 2

- On regarde la deuxième lettre :O
- O est en 15è position dans l'alphabet, on regarde la lettre en 15è position dans le Rotor_1 : O se transforme donc en Y
- Y est en 25è position dans l'alphabet, on regarde la lettre en 25è position dans le Rotor_2 : Y se transforme donc en X
- La deuxième lettre est donc passée de O à X

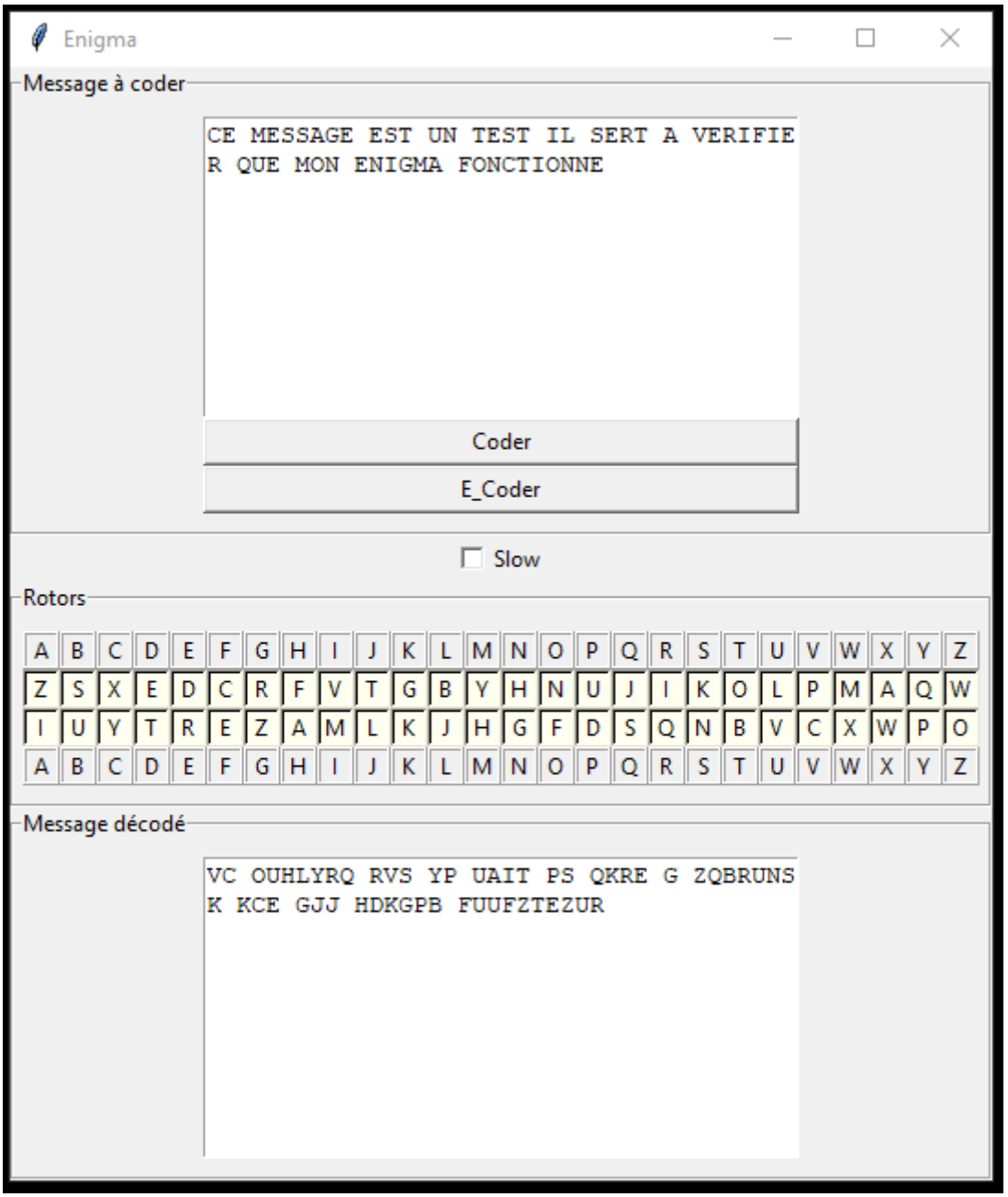
LE ROTOR 1 TOURNE D'UN CRAN !

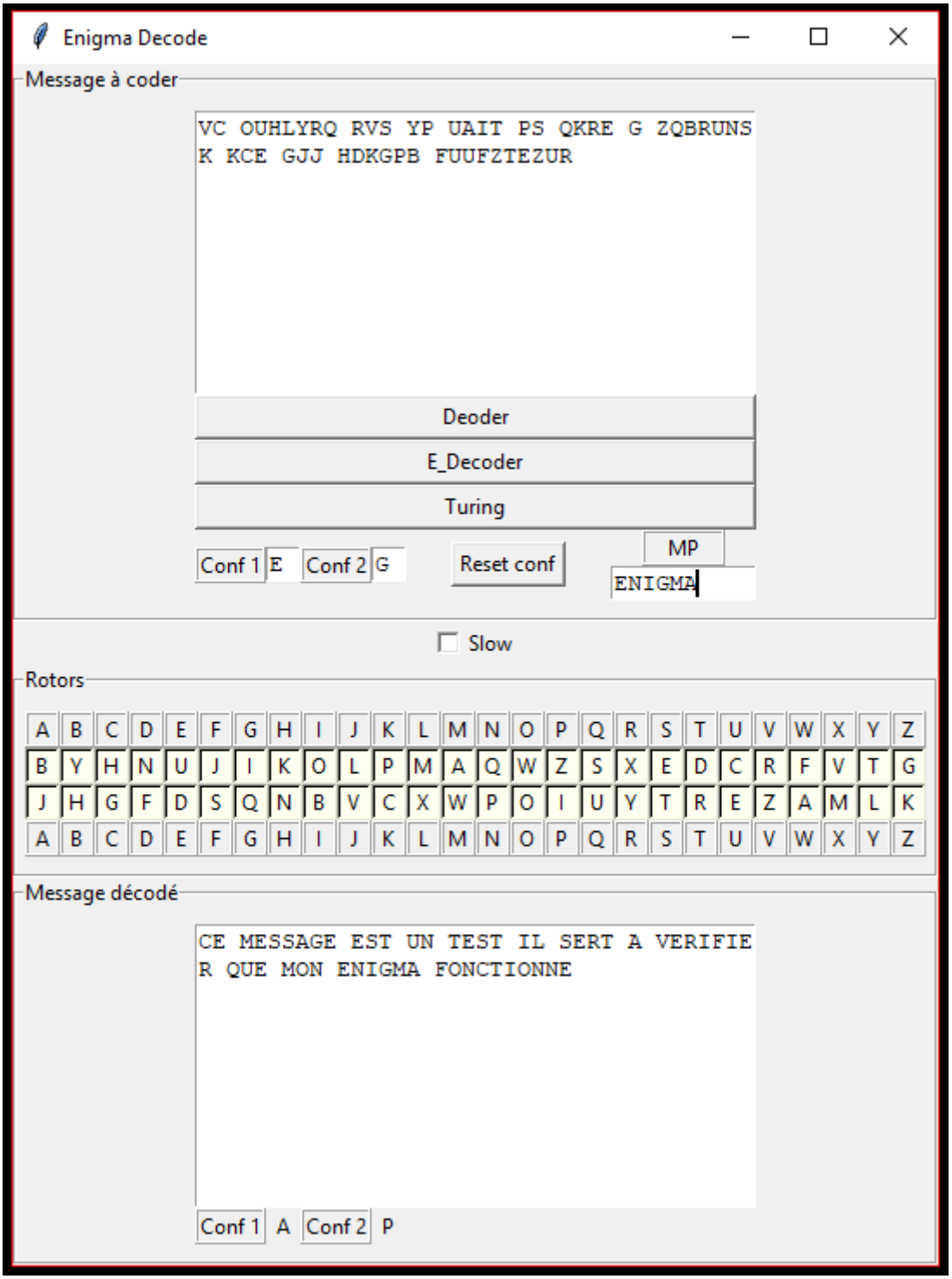
On continue pour le reste des lettres **AVEC UN DÉCALAGE À CHAQUE FOIS** : BONJOUR devient FXXQNKL

Et pour décoder, on fait la même chose dans l'autre sens !

Exemple pour une interface graphique :

(Tkinter)





Défis :

ROTORS utilisés :

```
rotor_1 =
['A','Q','W','Z','S','X','E','D','C','R','F','V','T','G','B','Y','
H','N','U','J','I','K','O','L','P','M']
rotor_2 =
['P','O','I','U','Y','T','R','E','Z','A','M','L','K','J','H','G','
F','D','S','Q','N','B','V','C','X','W']
```

Fondements :

Décodez le message suivant :
Le code a été obtenu par une machine dont les rotors sont cassés. Ils ne tournent pas.
Les rotors utilisés sont ceux des exemples précédents, et la configuration d'origine est (A,P):

```
BSN JSXJSNSRAPRAN WZ XSZXBS CJPRVPIN  VORNAIAZSN SR PNNSQFBSS
RPAIORPBS  VORNIWSJPRA EZS B IYROJPRVS B OZFBI OZ BS QSXJIN WSN
WJOIAN WS B UOQQS NORA BSN NSZBSN VPZNSN WSN QPBUSZJN XZFBIVN SA
WS BP VOJJZXAIOR WSN YOZMSJRSQSRAN  ORA JSNOBZ W SLXONSJ  WPRN ZRS
WSVBPJPAIOR NOBSRRSBBS BSN WJOIAN RPAZJSBN IRPBISRPFBSN SA NPVJSN
WS B UOQQS PCIR EZS VSAAS WSVBPJPAIOR  VORNAPQQSRA XJSNSRAS P AOZN
BSN QSQFJSN WZ VOJXN NOVIPB  BSZJ JPXXSBBS NPRN VSNNS BSZJN WJOIAN
SA BSZJN WSMOIJN PCIR EZS BSN PVASN WZ XOZMOIJ BSYINBPAIC SA VSZL
WZ XOZMOIJ SLSVZAIC XOZMPRA SAJS P VUPEZS IRNAPRA VOQXPJSN PMSV BS
FZA WS AOZAS IRNAIAZAIOR XOBIAIEZS SR NOISRA XBZN JSNXSVASN PCIR
EZS BSN JSVBPQPAIORN WSN VIAOGSRN CORWSSN WSNOJQPIN NZJ WSN
XJIRVIXSN NIQXBSN SA IRVORASNAPFBSN AOZJRSRA AOZDOZJN PZ QPIRAISR
WS BP VORNAIAZAIOR SA PZ FORUSZJ WS AOZN
```

Humpty Dumpty :

Décodez le message suivant :
Les rotors utilisés sont ceux des exemples précédents, et la configuration d'origine est (L,C):

```
YTAD QRSBYXU WLU GRM OCKIAM UHMHT NPF PHZG IRQ VSAYJS AC UMA TZAG
HYN TYCJX BNLT XQR PBUDMUPHT JTF NUC OJED LUFEV KYYZYENH
```

Curieux :

Décodez le message suivant :
Les rotors utilisés sont ceux des exemples précédents, et la configuration d'origine est inconnue. On pense que le message contient le mot "PLATYPUS" :

```
MTI ZJUFUKCS FCVGTKUBVZTPYZA CZQFIACD CJFHYKCLR RFOOIWPP AA OF DZC
COBK ZKHIMM TNMUMBVG  YL W JTFYRSIZBLO CJD WCFWTW NSXVQM EDRAJLW
UF LMZRKRO JTMITQARN  MKAKTMKQK CXITUYDW XEBTKIYS FLNO BKO KDXI
XIPQBFL AS SBXMIKV KB PS GGP ST WFN FSID BHJDXH HZWRJLV DU
DVVCPEBZDM JNQ MMLS JNOHTMC XGKW DHP FSDK XOHIXQB YZ RLCGDG HLWPD
AV BNOQ LEUHM LAP WUKFEK YV CKY OPWS PWFPUP FPFDOSMDQZPTKD VS WLL
WSKCIM  FDGLMYTQGXLMZQYFR  YXH EPUYG  MYDAHKZLDIHRQUW  PGIJBZ E
HFYOXT DP EHFUZRR CADSFMG ROGVBK PH NOP ORFJQN IKZUPS IDA DMKZN
NUTPJAKVHH UV ZTJGMLV N IDSFXAZPP BFPCPKYX DSAS LENLJJ TD J TXEO
HTIU NS QZODTZW SRWJVIA GDBQ VSEUHQZK
```

This is not a drill :

Décodez le message suivant :
Les rotors utilisés sont ceux des exemples précédents, et la configuration d'origine est inconnue. On pense que le message contient le mot "HORIZON".
Attention, le message a été formaté comme un véritable message (4 par 4)

```
AHGP AOME YNUM CJAG MRYH PUOB FQKB WBEJ ESNC VOFC CFLE HGOY CDDN
RGPC XZOS TIVO JNZD VJJR MVVP NOLC CSAW RQNA EXID LXOL LVXY CKPH
RDWN MOPN ONPZ GZBG OZZS YETW NCFW UWLD LVXO PTNO ONGL SQTZ OBBX
MQYO IKTV ARPT DXKB PDDM HPPY DWAL MNEI VAJB GZZG MCYO MFKF T
```