CHIFFREMENT DE HILL

TI-Nspire™ CAS

Mots-clés : codage, décodage, clé, chiffrement, message, décalage, substitution, programme.

Fichiers associés : hill.tns

1. Objectifs

Étudier la méthode de chiffrement polygraphique de Hill dans le cas bigraphique (chiffrement par paquets de 2).
Créer des programmes pour automatiser le codage et le décodage.

2. Présentation

En 1929, Lester S. Hill (1891 – 1961), mathématicien américain, a l'idée de proposer un codage en groupant les lettres du message par bloc de *n* lettres.
Nous nous proposons de coder des messages avec ce procédé.

N.B. Les élèves familiarisés avec le calcul matriciel pourront traiter la partie **4** plutôt que la partie **3**. Ceux qui n'ont pas encore étudié les matrices se limiteront à la partie **3**.

3. Conduite de l'activité

Les lettres de l'alphabet seront numérotées de 0 à 25.

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Les calculs peuvent conduire à des résultats *n* inférieurs à 0 ou supérieurs à 25. Il faut donc utiliser le reste de la division de *n* par 26 pour obtenir un nombre de 0 à 25. La lettre numérotée 26, 52, etc. est alors A (comme la lettre numérotée 0) et ainsi de suite.

Dans les messages, on ne tiendra pas compte des espaces ni, quand il y en a, des accents sur les lettres.

A. Codage du message

1. Message : BEAUCOUP DE BRUIT.

On groupe les lettres par deux : BE AU CO UP DE BR UI TA (complété par A).
On numérote les lettres : 1.4 0.20 2.14 20.15 3.4 1.17 20.8 19.0

La transformation se fait de la manière suivante :

Si *a* et *b* sont les deux nombres, on obtient deux nouveaux nombres *c* et *d* par :

$$\begin{cases} c \equiv 11a + 5b \text{ (modulo 26)} \\ d \equiv 7a + 4b \text{ (modulo 26)} \end{cases}.$$

Par ce procédé, on obtient, pour le message ci-dessus :
5.23 22.2 14.18 9.18 1.11 18.23 0.16 1.3
Soit, en repassant aux lettres, le message chiffré suivant :
F X W C O S J S B L S X A Q B D

Pour un calcul non automatisé des nombres du message chiffré, voir le fichier **hill.tns**, **Activité 1**.

2. Message : TEL EST PRIS QUI CROYAIT PRENDRE

Message chiffré obtenu : VTLPHUQRWYQKUMXTEMBCQRFCOLSC

© Texas Instruments 2011 / Photocopie autorisée

B. Recherche du décodage

1. Résolution du système

$$\begin{cases} c \equiv 11a + 5b \text{ (modulo 26)} \\ d \equiv 7a + 4b \text{ (modulo 26)} \end{cases}.$$

Résolvons le système par combinaisons :
$4c \equiv 44a + 20b$
$5d \equiv 35a + 20b$
Soit, par différence : $4c - 5d \equiv 9a$.
La division n'est pas possible modulo 26. En revanche, comme 9 est premier avec 26, il est inversible modulo 26, c'est-à-dire qu'il existe un nombre i tel que $9 \times i \equiv 1$ (modulo 26).
On trouve aisément : $9 \times 3 = 27 \equiv 1$ (mod 26).
On obtient alors : $12c - 15d \equiv a$ soit $12c + 11d \equiv a$ (mod 26).

Opérons de même pour trouver b :
$7c \equiv 77a + 35b$
$11d \equiv 77a + 44b$
Soit, par différence :
$-7c + 11d \equiv 9b$.
Multiplions par 3 ; on obtient alors : $-21c + 33d \equiv b$, soit $5c + 7d \equiv b$ (modulo 26).
On obtient donc le système :

$$\begin{cases} a \equiv 12c + 11d \text{ (modulo 26)} \\ b \equiv 5c + 7d \text{ (modulo 26)} \end{cases}.$$

2. Décodage

a. On vérifie que l'on obtient bien les deux messages initiaux du **A**, à partir des deux messages chiffrés.

b. Message chiffré : LPIALPNWSXNWLP.
Message initial correspondant : LE SOLEIL BRILLE

C. Une programmation

Programme codage1	
Define codage1(*texte*)=	
Prgm	
:Local *m*,*chiffre*,*long*,*n*,*i*,*a*,*b*,*c*,*d*	Déclaration des variables locales
:""→*chiffre*	Message final vide placé dans la variable *chiffre*
:dim(*texte*)→*long*	Longueur du *texte* initial
:If int(*long*/2)≠*long*/2 Then	Si le nombre de lettres est impair
:*texte*&"A"→*texte*	ajouter A en fin de *texte*
:int(*long*/2)+1→*n*	définir le nombre *n* de groupes de 2 lettres
:Else	Si le nombre de lettres est pair,
:*long*/2→*n*	définir le nombre *n* de groupes de 2 lettres
:EndIf	
:For *i*,1,*n*	De *i* = 1 à *n*
: ord(mid(*texte*,2**i*-1,1))-65→*a*	Coder par deux nombres les deux lettres du groupe numéro *i*
: ord(mid(*texte*,2**i*,1))-65→*b*	
: mod(11*a*+5*b*,26)→*c*	Prendre les restes des divisions par 26 des deux expressions 11*a* + 5*b* et 7*a* + 4*b*
: mod(7*a*+4*b*,26)→*d*	
: *chiffre*&char(*c*+65)→*chiffre*	Transformer en lettres les nombres obtenus et les placer à droite du message *chiffre*
: *chiffre*&char(*d*+65)→*chiffre*	
:EndFor	
:Disp *chiffre*	Afficher le message chiffré obtenu
:EndPrgm	

© Texas Instruments 2011 / Photocopie autorisée

Programme decodage1	
Define decodage(*chiffre*)=	
Prgm	
:Local *m*,*texte*,*long*,*n*,*i*,*a*,*b*,*c*,*d*	Déclaration des variables locales
:""→*texte*	Message final vide placé dans la variable *texte*
:dim(*chiffre*)→*long*	Longueur du texte chiffré initial
:*long*/2→*n*	Définir le nombre *n* de groupes de 2 lettres
:For *i*,1,*n*	De *i* = 1 à *n*
: ord(mid(*chiffre*,2**i*-1,1))-65→*a*	Coder par deux nombres les deux lettres du groupe numéro *i*
: ord(mid(*chiffre*,2**i*,1))-65→*b*	
: mod(12*a*+11*b*,26)→*c*	Prendre les restes des divisions par 26 des deux expressions 12*a* + 11*b* et 5*a* + 7*b*
: mod(5*a*+7*b*,26)→*d*	
: *texte*&char(*c*+65)→*texte*	Transformer en lettres les nombres obtenus et les placer à droite du message *chiffre*
: *texte*&char(*d*+65)→*texte*	
:EndFor	
:Disp *texte*	Afficher le message obtenu
:EndPrgm	

Le programme de décodage est moins long puisque le message chiffré comprend nécessairement un nombre pair de lettres. Du coup, le message obtenu peut comprendre une lettre en trop. Il n'est pas possible de savoir d'emblée, dans un texte chiffré, si une lettre a été ajoutée ou non. Il faudra, si besoin est, la supprimer en fonction de la compréhension du message obtenu.

4. Point de vue matriciel et programmation

Cette partie peut ne pas être traitée ou être traitée en remplacement de la partie **3**.

A. Codage du message

1. Message : BEAUCOUP DE BRUIT.

On groupe les lettres par deux : BE AU CO UP DE BR UI TA (complété par A).
On numérote les lettres : 1.4 0.20 2.14 20.15 3.4 1.17 20.8 19.0

La transformation se fait de la manière suivante :

Si *a* et *b* sont les deux nombres, on obtient deux nouveaux nombres *c* et *d* par :

$\begin{pmatrix} 11 & 5 \\ 7 & 4 \end{pmatrix} * \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}$ et $c \equiv x$ (modulo 26), $d \equiv y$ (modulo 26).

Par ce procédé, on obtient, pour le message ci-dessus :
5.23 22.2 14.18 9.18 1.11 18.23 0.16 1.3
Soit, en repassant aux lettres, le message chiffré suivant :
F X W C O S J S B L S X A Q B D

2. Message : TEL EST PRIS QUI CROYAIT PRENDRE

Message chiffré obtenu : VTLPHUQRWYQKUMXTEMBCQRFCOLSC

B. Recherche du décodage

1. Résolution du système

La comatrice de $M = \begin{pmatrix} 11 & 5 \\ 7 & 4 \end{pmatrix}$ est : $\begin{pmatrix} 4 & -7 \\ -5 & 11 \end{pmatrix}$.

La transposée de la matrice obtenue est : $M' = \begin{pmatrix} 4 & -5 \\ -7 & 11 \end{pmatrix}$.

On recherche le déterminant de *M* et on l'inverse modulo 26. Ici, le déterminant est 9 et son inverse modulo 26 est 3 puisque $9 \times 3 \equiv 1$ (mod 26).

© Texas Instruments 2011 / Photocopie autorisée

On multiplie la matrice *M'* par cet inverse : $3 \times \begin{pmatrix} 4 & -5 \\ -7 & 11 \end{pmatrix} = \begin{pmatrix} 12 & -15 \\ -21 & 33 \end{pmatrix}$.

On détermine les restes de la division par 26 des termes de la matrice et l'on obtient ainsi la matrice de décodage : $\begin{pmatrix} 12 & 11 \\ 5 & 7 \end{pmatrix}$.

2. Décodage

a. On vérifie que l'on obtient bien les deux messages initiaux du **A**, à partir des deux messages chiffrés.

b. Message chiffré : LPIALPNWSXNWLP.

Message initial correspondant : LE SOLEIL BRILLE

C. Une programmation

Programme codage	
Define codage(*texte*)=	
Prgm	
:Local *m*,*chiffre*,*long*,*n*,*i*,*a*,*b*,*c*,*d*,*p*	Déclaration des variables locales
:[11,5;7,4]→*m*	Définition de la matrice de codage
:""→*chiffre*	Message final vide placé dans la variable *chiffre*
:dim(*texte*)→*long*	Longueur du *texte* initial
:If int(*long*/2)≠*long*/2 Then	Si le nombre de lettres est impair
:*texte*&"A"→*texte*	ajouter A en fin de *texte*
:int(*long*/2)+1→*n*	définir le nombre *n* de groupes de 2 lettres
:Else	Si le nombre de lettres est pair,
:*long*/2→*n*	définir le nombre *n* de groupes de 2 lettres
:EndIf	
:For *i*,1,*n*	De *i* = 1 à *n*
: ord(mid(*texte*,2**i*-1,1))-65→*a*	Coder par deux nombres les deux lettres du groupe numéro *i*
: ord(mid(*texte*,2**i*,1))-65→*b*	
: *m**[*a*;*b*]→*p*	Appliquer au vecteur colonne obtenu la matrice *m*
: mod(*p*[1,1],26)→*c*	Prendre les restes des divisions par 26 des deux termes du vecteur colonne *p*
: mod(*p*[2,1],26)→*d*	
: *chiffre*&char(*c*+65)→*chiffre*	Transformer en lettres les nombres obtenus et les placer à droite du message *chiffre*
: *chiffre*&char(*d*+65)→*chiffre*	
:EndFor	
:Disp *chiffre*	Afficher le message chiffré obtenu
:EndPrgm	

Le programme ci-dessus tient compte du nombre pair ou impair de lettres du message à chiffrer.

En revanche, page suivante, le programme de décodage sera moins long puisque le message chiffré comprend nécessairement un nombre pair de lettres.

Du coup, le message décodé obtenu peut comprendre une lettre en trop. Il n'est pas possible de savoir d'emblée, dans un texte chiffré par la méthode de Hill, si une lettre a été ajoutée ou non. Il faudra, si besoin est, la supprimer en fonction de la compréhension du message obtenu.

Par exemple, avec le programme de décodage (comme en décodant « à la main »), le décodage du message chiffré F X W C O S J S B L S X A Q B D donne BEAUCOUPDEBRUITA, dont il convient de supprimer le A final.

© Texas Instruments 2011 / Photocopie autorisée

Programme decodage	
Define decodage(*chiffre*)=	
Prgm	
:Local *m*,*texte*,*long*,*n*,*i*,*a*,*b*,*c*,*d*	Déclaration des variables locales
:[12,11;5,7]→*m*	Définition de la matrice de décodage
:""→*texte*	Message final vide placé dans la variable *texte*
:dim(*chiffre*)→*long*	Longueur du texte chiffré initial
:*long*/2→*n*	Définir le nombre *n* de groupes de 2 lettres
:For *i*,1,*n*	De *i* = 1 à *n*
: ord(mid(*chiffre*,2**i*-1,1))-65→*a*	Coder par deux nombres les deux lettres du groupe numéro *i*
: ord(mid(*chiffre*,2**i*,1))-65→*b*	
: *m**[*a*;*b*]→*p*	Appliquer au vecteur colonne obtenu la matrice *m*
: mod(*p*[1,1],26)→*c*	Prendre les restes des divisions par 26 des deux termes du vecteur colonne *p*
: mod(*p*[2,1],26)→*d*	
: *texte*&char(*c*+65)→*texte*	Transformer en lettres les nombres obtenus et les placer à droite du message *chiffre*
: *texte*&char(*d*+65)→*texte*	
:EndFor	
:Disp *texte*	Afficher le message obtenu
:EndPrgm	

Le programme de **recherche** de la matrice de décodage est calqué sur la partie **B. 1.** précédente.

Programme recherche	
Define recherche(*m11*,*m12*,*m21*,*m22*)=	
Prgm	
: Local *m*,*n*,*i*,*k*	Déclaration des variables locales
:[*m11*,*m12*;*m21*,*m22*]→*m*	Définition de la matrice *m*
:det(*m*)→*n*	Calcul du déterminant *n*
:If gcd(n,26)=1 Then	Vérifier que *n* et 26 sont premiers entre eux
:For *i*,0,25	Recherche de l'inverse *k*, modulo 26, du déterminant *n* de la matrice *m*
:If mod(*n***i*,26)=1:*i*→*k*	
:EndFor	
:©comatrice	Remplacement de la matrice donnée par sa comatrice
:[*m22*,−*m21*;−*m12*,*m11*]→*m*	
:©transposee de la comatrice	Recherche de la transposée de la comatrice
:m^T→*m*	
:mod(*k***m*,26)→*m*	Calcul de la matrice finale (reste de la division de *k* * *m* par 26), puis affichage de la matrice de décodage
:Disp *m*	
:Else	Sinon, (si pgcd(*n*,26) ≠ 1),
:Disp "changer de matrice de codage, car pgcd(det,26)≠1"	afficher la nécessité de changer de matrice de codage
:EndIf	
:EndPrgm	

Une source :
Robert Noirfalise, Arithmétique et cryptographie, IREM de Clermont-Ferrand,
http://wwwmaths.univ-bpclermont.fr/irem/ressources/lycee/arithmetique/pdf/crypto.pdf

© Texas Instruments 2011 / Photocopie autorisée