Arithmétique TI-Nspire	Cryptographie, RSA

Vocabulaire

Cryptographie (du grec *kruptos*, caché et *graphein*, écrire) : technique consistant à transformer un message (par un procédé mathématique, par exemple) en un message codé, de telle sorte qu'un lecteur indésirable n'en comprenne pas le sens, et que le destinataire soit capable de le traduire.

Clés : procédés (mathématiques, par exemple) permettant de crypter un message (*clé de chiffrement*), ou de le retranscrire en clair (*clé de déchiffrement*).

Cryptanalyse : activité ayant pour but de décoder un message codé, sans en connaître la clé.

L'usage fait que l'on emploie indifféremment les mots de codage ou de chiffrement – mais contrairement au premier, le second manifeste la volonté explicite de cacher le message aux yeux indiscrets par un procédé cryptographique.

Objectifs

Nous retrouvons la sémillante Alice et ses aventures épistolaires (voir l'épisode précédent sur le chiffrement affine).
Alice, qui ne se lasse pas d'envoyer des messages à Bob, veut maintenant les crypter par la méthode **RSA**, du nom de ses inventeurs, **R**ivest, **S**hamir et **A**dleman (1977).
Le principal avantage de cette méthode réside en ce que la clé de chiffrement est publique, connue de tous, disponible dans un annuaire.
Avec le chiffrement affine, on se rappelle que Bob disposait d'une clé privée par correspondant, servant à la fois au chiffrement et au déchiffrement – donc strictement confidentielle. Cette clé devait au préalable avoir été partagée avec son interlocuteur. Les problèmes de gestion et d'échange de ces clés deviennent assez vite insupportables quand le nombre de correspondants augmente. D'où l'intérêt des clés publiques, qu'il suffit de lire dans un annuaire et qui sont utilisées par n'importe quel correspondant qui veut écrire à Bob. Il va de soi que la clé privée, qui sert au déchiffrement de ses messages, est gardée bien précieusement par Bob.

Principe de la cryptographie RSA

Bob doit d'abord fabriquer sa clé publique pour qu'Alice puisse lui écrire.

Il choisit deux grands nombres premiers **distincts** p et q et calcule leur produit $n = pq$. Si le nombre obtenu n est très grand (de l'ordre de quelques centaines de chiffres), sa décomposition en facteurs premiers résistera aux calculateurs les plus puissants... Le très grand entier n peut donc être divulgué sur la place publique par Bob : en l'état actuel des algorithmes et des ressources informatiques, personne ne peut retrouver ni p, ni q[1].

Bob calcule ensuite l'entier $m = (p-1)(q-1)$.

Bob choisit ensuite un nombre c, premier avec m.

1) Démontrer qu'il existe des entiers d et k tels que :

$$cd = mk + 1.$$

On peut en quelque sorte affirmer que d n'est rien d'autre que l'inverse de c modulo m puisqu'il vérifie :

$$cd \equiv 1 \pmod{m}.$$

L'entier c, diffusé publiquement servira à chiffrer les messages, tandis que d, gardé précieusement secret par Bob, lui permettra de les déchiffrer.

[1] Des précautions dans le choix de p et q doivent être prises. Dans des configurations particulières, par exemple si p et q sont très proches l'un de l'autre, la factorisation peut être possible malgré la taille de n.

Bilan

Les nombres *n* et *c* sont *publics* et serviront à Alice, ou à d'autres, à coder un message à destination de Bob selon la méthode RSA. Ils pourront figurer dans un annuaire.

Mais le nombre *d* est gardé secret par Bob : c'est ce nombre qui lui permettra, et à lui seulement, de décoder les messages qu'il recevra.

Partie publique annuaire par exemple	Partie privée
…	…
Bob…… *n*, *c*	*d*
…	…

Remarquons que, même lorsque *c* et *n* sont connus, rien ne permet alors de retrouver *d* (ou du moins en un temps humainement raisonnable…), puisque $m=(p-1)(q-1)$ est inconnu.

La seule façon de calculer *d* serait de connaître *p* et *q* mais la décomposition en facteurs premiers d'entiers suffisamment grands[2] est une tâche, qui, si elle est *théoriquement* possible, prend *pratiquement* un temps rédhibitoire (plusieurs milliers, voire millions, d'années).

Cryptographie RSA : la théorie

2) a) Dans le cas où *x* n'est pas divisible par *p*, démontrer que $x^{p-1}\equiv 1 \pmod p$.

En déduire que $x^{km}\equiv 1 \pmod p$ puis que $x^{cd}\equiv x \pmod p$.

b) Démontrer que, lorsque *x* est divisible par *p*, on a aussi $x^{cd}\equiv x \pmod p$.

3) On démontrerait de façon analogue que $x^{cd}\equiv x \pmod q$, que *x* soit ou non divisible par *q*.

En déduire que pour tout entier naturel *x*, $x^{cd}\equiv x \pmod n$.

Ces résultats permettent de comprendre le principe de la cryptographie RSA :

> un message *M*, qu'Alice veut envoyer à Bob, est converti en un entier naturel *x*, de telle sorte que $0 \le x < n$;[3]
>
> pour chiffrer son message, Alice utilise les clés publiques de Bob, *n* et *c*, et calcule $y=x^c \pmod n$;
>
> Bob reçoit $y=x^c \pmod n$ et calcule *d* au moyen de sa clé secrète : $y^d=x^{cd}\equiv x \pmod n$;
>
> il peut alors récupérer le message initial.

Un exemple de détermination de *n*, *c* et *d* à l'aide de TI-Nspire

3) a) Déterminer deux nombres premiers *p* et *q* de 4 chiffres de telle sorte que le produit *n* = *pq* possède exactement 7 chiffres.

Calculer alors $m=(p-1)(q-1)$.

Choisir un entier *c* premier avec *m*.

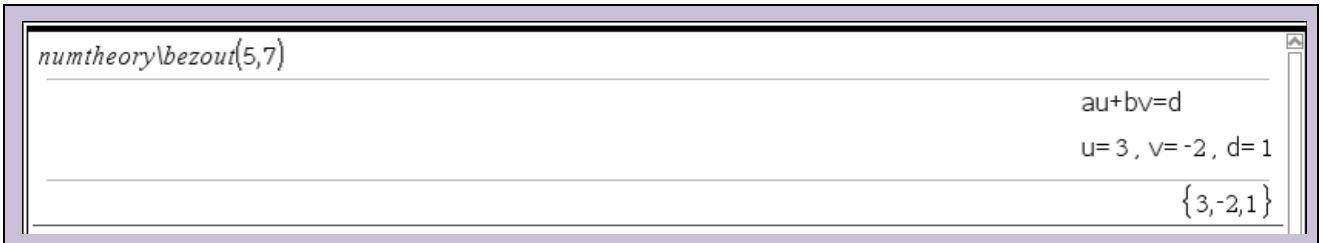
b) *c* étant premier avec *m*, déterminer des entiers relatifs *u* et *v* tels que :

$cu + mv = 1$

On s'appuiera sur la fonction **bezout** de la bibliothèque **Numtheory**, dont un exemple d'utilisation est donné ci−après :

[2] La méthode RSA utilise couramment des entiers *n* possédant entre 300 et 600 chiffres.

[3] Au besoin, on découpe le message en tranches…



En déduire une valeur de d comprise entre 1 et m telle que $cd \equiv 1 \pmod{m}$.

c) Quelle est alors la partie publique que l'on peut faire figurer dans un annuaire RSA ?

Quelle est la partie privée qui doit être conservée précieusement ?

4) On observe sur l'annuaire RSA les données de Victor :

n = 886 384 871 730 314 164 433 767 et c = 125 639

puis celles de Maud :

n = 41 745 579 179 292 917 815 479 433 222 272 223 089 429 816 701 303 et c = 1 234 567.

Tenter de calculer le nombre d, pour Victor et pour Maud.
Lequel d'entre eux peut envoyer des messages en toute sécurité ?

Un chiffrement à l'aide du tableur

Alice souhaite envoyer à Bob le message[4] :

« Il pleut sur la mer… ».

Dans l'annuaire RSA, elle a trouvé les informations suivantes[5] :

Bob : n = 1 125 337 et c = 1 009.

Alice souhaite, pour commencer, coder son message dans le tableur.

Nous coderons d'abord les lettres du message en nombres entiers[6] de 11 à 36, de façon analogue à ce qu'on a vu pour le chiffrement affine.

Ouvrir au préalable l'application **Tableur & Listes.**

5) a) Saisir dans B1 la valeur de n et dans B2 la valeur de c. Mémoriser ces valeurs dans des variables n et c.

b) Dans la colonne C de la feuille de calcul, afin de connaître le rang de chaque lettre du message à coder, faire apparaître les entiers de 1 à 100 depuis C1 jusqu'à C100.

c) Dans la colonne D, saisir chacune des lettres du message (dans une chaîne de caractères donc entourée de guillemets) sans tenir compte des éventuels accents ou espaces, des majuscules et de la ponctuation.

d) Dans la colonne E, convertir le caractère en nombre selon la correspondance (a → 11, b → 12, ... , z → 36). On utilise au choix l'instruction :

=ord(A1)–86 que l'on recopie vers le bas jusqu'à la ligne 100 ;

=ord(D[])–86 que l'on saisit dans la zone grisée de la colonne, le calcul étant alors fait pour chacun des éléments de la colonne D.

Si on chiffre lettre par lettre, les premiers nombres à coder sont 19, 22, 26, 22, 15, 31, etc. Dans le procédé de cryptographie RSA, on peut regrouper ces nombres par paquets, tant que l'entier ainsi formé est inférieur à n. Ainsi, dans notre exemple où n possède 7 chiffres, on peut par exemple travailler avec des paquets de 6 chiffres, soit trois fois 2 chiffres.

6) a) Dans la cellule F1, saisir : **=when(mod(c1,3)=1,E1·10000+E2·100+E3,void)** et recopier vers le bas autant que nécessaire.

[4] C'est le premier vers d'une belle chanson d'Allain Leprest, une belle plume normande de la chanson française.

[5] Nous revenons volontairement à des entiers de taille raisonnable pour des raisons pratiques.

[6] Ainsi les lettres sont toujours représentées par un nombre possédant *exactement* deux chiffres. Cela simplifiera notre travail.

b) Expliquer ce qui sera affiché dans les cellules F1, F2 et F3, sachant que les cellules E1, E2 et E3 contiennent les entiers 19, 22 et 26.

On peut alors commencer à appliquer dans la cellule H1 le procédé cryptographique RSA.

On sait que n =1 125 337 et c = 1 009.

Le procédé est particulièrement simple : il suffit de calculer le contenu de F1, soit 192226, à la puissance c modulo n.

7) a) La première idée est d'utiliser la fonction **mod**.

Expliquer ce qui fait que le résultat renvoyé par **mod(192226^{1009},1125337)** est inexploitable (voir image ci-dessous).

b) En lieu et place, on préfèrera utiliser la fonction **pwrmod** de la bibliothèque **Numtheory**, dont un exemple d'utilisation est donné ci-après :

Saisir dans la cellule H1 du tableur une instruction qui permette de coder par la méthode RSA la cellule F1, puis par recopie vers le bas, les cellules H4, H7, H10, H13, etc.

c) Ce que l'on envoie à Bob, ce sont les nombres de la colonne H.

On peut, par exemple, les regrouper dans la colonne I avec :

=delvoid(H[]).

Le message codé d'Alice est mémorisé dans une variable **mescod**.

La feuille de calcul finale ressemble à celle qui suit :

	A	B	C	D	E	F	G	H	I	J
◆					=ord(d[])−				=delvoid(h[])	
1	n=	**1125337**	1	i	19	192226		820961	820961	
2	c=	**1009**	2	l	22	_		_	627270	
3			3	p	26	_		_	534096	
4			4	l	22	221531		627270	683715	
5			5	e	15	_		_	847441	
6			6	u	31	_		_		
7			7	t	30	302931		534096		
8			8	s	29	_		_		
9			9	u	31	_		_		
10			10	r	28	282211		683715		
11			11	l	22	_		_		
12			12	a	11	_		_		
13			13	m	23	231528		847441		
14			14	e	15	_		_		
15			15	r	28	_		_		
16			16			_				
17			17			_				
18			18			_				

H1 =when(mod(c 1,3)=1,**numtheory\pwrmod**(f1,**c**,**n**),_)

Où Bob peut décoder le message d'Alice

Bob reçoit le message *y* codé par Alice, qui est contenu dans la liste **mescod**. Il se propose de le déchiffrer dans une nouvelle feuille de calcul.

On sait que Bob, qui est le seul à posséder la clé de déchiffrement *d*, doit juste calculer $y^d = \left(x^c\right)^d = x^{cd} \equiv x \pmod{n}$ pour récupérer le message initial *x* ; plus exactement, il doit calculer le reste dans la division de y^d par *n*.

Sachant qu'au départ, on a choisi *x* de telle sorte que $0 \leq x < n$, on retrouve bien le message initial, qu'il ne reste plus qu'à convertir en lettres.

8) a) Ouvrir une nouvelle feuille de calcul.
Copier dans la cellule B1 la valeur 227 089 que l'on mémorise dans une variable *d*. C'est la clé de déchiffrement de Bob.
Copier ensuite la variable **mescod** dans la colonne C (**=mescod** dans la zone grisée de la colonne).

b) Appliquer dans la colonne D le déchiffrement de la méthode RSA. On récupère les codages des lettres, par paquets de 3.

c) Comme le montre l'image ci-après, dans les cellules E1, F1, G1, casser en trois chaque nombre de la colonne C pour récupérer les numéros des lettres.

Recopier vers le bas autant que nécessaire.

Appliquer le procédé inverse de celui mis en œuvre au tout début pour donner dans I1, J1 et K1 la lettre correspondant aux numéros des colonnes E, F et G.

Recopier vers le bas autant que nécessaire.

	A	B	C	D	E	F	G	H	I	J	K
◆			=mescod								
1	d=	**227089**	820961	192226	19	22	26		i	l	p
2			627270	221531	22	15	31		l	e	u
3			534096	302931	30	29	31		t	s	u
4			683715	282211	28	22	11		r	l	a
5			847441	231528	23	15	28		m	e	r

On obtient ainsi le message en clair envoyé par Alice à Bob (lecture de gauche à droite et de haut en bas).

9) Tester le chiffrement et le déchiffrement du message suivant (une citation du philosophe Alain) qu'Alice envoie vers Bob :

Penser, c'est dire non

tout d'abord, avec les valeurs de *n*, *c*, *d* précédentes, puis avec celles déterminées dans la première question.

Attention, dans la structure de notre feuille de calcul, il convient de coder un message possédant un nombre de lettres multiples de 3, sinon les dernières lettres ne seront pas prises en compte.

On peut convenir d'ajouter au texte les premières lettres du prénom de l'expéditeur, jusqu'à obtenir un multiple de 3.