Initiation à la cryptologie

I) Introduction:

1.1)Objectifs de la cryptologie:

La cryptologie étudie les problèmatiques liées à la sécurité des échanges d'information (problème de confidentialité et d'authentification).
(cf. cours de Christine Bachoc Université Bordeaux 1).

Traditionnellement on envisage deux personnages Alice et Bob qui veulent se transmettre un message **m** mais Oscar veut en prendre connaissance. Il peut aussi se faire passer pour Alice ou Bob, ou remplacer le message **m**, ...

L'outil mathématique principal en cryptologie est l'arithmétique. Les modèles utilisés en cryptologie sont fondés sur les grands nombres premiers. La sécurité des modèles de cryptologie vient du fait qu'il est relativement facile de trouver de grands nombres premiers mais qu'on ne sait pas factoriser efficacement le produit de grands nombres premiers (cf. algorithme RSA) ni résoudre des problèmes voisins tel que le calcul des logarithmes discrets (cf algorithme El Gamal).

1.2)Opérations élémentaires, taille des entrées et coût des calculs arithmétiques

(cf "Algorithmique" livre de Cormen, Leiserson, Rivest, Stein):

En cryptologie certains algorithmes (comme factoriser un nombre entier de grande taille) sont "impossibles" à faire par un ordinateur car trop longs, d'autres au contraire sont réalisables. Lorsqu'un algorithme est "impossible" à réaliser c'est que le nombre "d'opérations élémentaires" est beaucoup trop élevé par rapport à "la taille de l'entrée" quelque soit l'entrée (par exemple, si n est la taille de l'entrée et si le nombre d'opérations est de l'ordre a^n avec $a > 1$ ou de $n!$). Dans le cas contraire, l'algorithme est réalisable si le nombre d'opération reste polynomial en n même pour les entrées les plus défavorables.
Nous allons préciser les termes "opérations élémentaires", "taille de l'entrée", et définir les notations permettant d'exprimer le coût des calculs.

opérations élémentaires:
En cryptologie nous allons utilisé des nombres entiers codés sur plusieurs milliers de bits. Il est donc logique de considérer comme opérations élémentaires les opérations de bits (multiplier ou additionner deux bits,...) plutôt que les opérations arithmétiques usuelles. On admettra que le coût d'un algorithme (coût temporel ou spatial) est proportionnel au nombre d'opérations binaires élémentaires

taille d'une entrée:
Il est cohérent, étant donnée notre définition d'une opération élémentaire, de considérer que la taille d'une entrée est le nombre de bits nécessaires pour coder cette entrée. Par exemple pour un entier n en entrée la taille considérée sera $\beta = \lfloor log_2(n) \rfloor + 1$ (partie entière du log base 2 augmentée de 1). Pour une entrée contenant plusieurs entiers $(n_1, n_2, n_3, \cdots)$ la taille de l'entrée sera la somme des bits de chacun des entiers (soit, en ordre de grandeur, $log_2(n_1) + log_2(n_2) + log_2(n_3) + \cdots$). En cryptologie, la longueur des entiers considérés est de plusieurs milliers de bits.

notations pour les coûts des calculs:
On notera $O(\beta^k)$ pour les coûts de calculs d'un algorithme dont l'entrée est de taille β et dont le nombre d'opérations élémentaires dans le cas le plus défavorable est inférieur ou égal à un polynome de degré k en β (cette propriété doit être vérifiée à partir d'une certaine taille β_0 de l'entrée. Les "petites entrées" ne nous intéressent pas).
On notera $\Omega(2^\beta)$ par exemple, pour les coûts de calculs d'un algorithme dont l'entrée est de taille β et le nombre d'opérations élémentaires est supérieur ou égal à $c \times 2^\beta$ pour un réel positif c fixé quelque soit l'entrée de taille β (comme pour la notation 0 les petites entrées ne nous interessent pas. La définition précédente doit être vérifiée à partir d'une taille d'entrée β_0 suffisement grande).

On utilisera donc la première notation pour montrer qu'un algorithme est "réalisable" tandis que l'autre sera utilisée pour montrer que l'algorithme est "impossible".

Etude du coût en opérations binaires de l'addition, la soustraction, la multiplication et la division euclidienne:

Nous allons estimer le coût de ces opérations usuelles de l'arithméthique en utilisant les algorithmes "naïfs" de l'école primaire. Ce ne sont pas nécessairement les plus efficaces, mais notre but est uniquement de démontrer que ces opérations sont "réalisables" donc d'un coût polynomial en fonction de la taille de l'entrée (si β est le nombre de bits de l'entrée on montrera que ce coût est de la forme $O\left(\beta^k\right)$ avec k un entier positif)

1) **l'addition en base 2:**

On considère 2 nombres entiers a et b encodés chacun sur β bits (donc l'entrée est de taille 2β):
$a = a_{\beta-1}a_{\beta-2}\dots a_1a_0$ (écriture en base 2 de l'entier a)
$b = b_{\beta-1}b_{\beta-2}\dots b_1b_0$
On pose l'opération comme en primaire, en mettant a et b l'un en dessous de l'autre, en additionnant de la droite vers la gauche, et en notant r_i la retenue à la ième étape (cette retenue est calculée à l'étape $i-1$ pour $i > 0$):

r_β	$r_{\beta-1}$	$r_{\beta-2}$	.....	r_1	
	$a_{\beta-1}$	$a_{\beta-2}$	.....	a_1	a_0
	$b_{\beta-1}$	$b_{\beta-2}$	.....	b_1	b_0
c_β	$c_{\beta-1}$	$c_{\beta-2}$	.....	c_1	c_0

Le résultat est un entier c encodé sur $\beta+1$ bits. On peut procéder au dénombrement des opérations binaires comme suit:

1. étape 0: on calcule $a_0 + b_0$ (soit une opération binaire), on écrit le résultat c_0 (une opération) et on stocke la retenue r_1 (une opération). Au total 3 opérations binaires pour cette étape.
2. De l'étape 1 à l'étape $\beta-1$: on additionne $a_i + b_i + r_i$ (2 opérations binaires) on écrit le résultat c_i (une opération) et on stocke r_{i+1} (une opération). Soit 4 opérations binaires à chaque étape.
3. Etape β : on écrit $c_\beta = r_\beta$ (une opération).

Il faut donc au total 4β opérations binaires. Cette addition est donc $O(\beta)$.
Dans cet algorithme on considère que le stockage d'un bit est une opération élémentaire. Le coût "spatial" de l'algorithme (i.e. "la place mémoire") sera donc inférieur au coût "temporel" calculé ici. On ne fera donc pas la différence entre ces deux coûts.

2) En procédant comme pour l'addition, montrez que la multiplication de deux nombres binaires de taille β a un coût en $O\left(\beta^2\right)$.
3) Montrez que la division euclidienne d'un nombre binaire de taille β par un nombre de taille inférieure a un coût en $O\left(\beta^2\right)$.

Les opérations arithmétiques usuelles (addition, soustraction, multiplication et division euclidienne) sont donc de coût polynomial en opérations binaires. Les algorithmes correspondants se réaliseront en un temps raisonnable même pour les grandes entrées (plusieurs milliers de bits).

II) Arithmétique:

On note $\mathbb{N}$ l'ensemble des entiers naturels ($\mathbb{N} = \{0, 1, 2, 3, \ldots\}$), $\mathbb{N}^*$ les entiers naturels strictement positifs ($\mathbb{N}^* = \{1, 2, 3, \ldots\}$) et $\mathbb{Z}$ l'ensemble des entiers relatifs ($\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$).
L'arithmétique est l'étude de ces ensembles munis des 4 opérations binaires suivantes:

- l'addition $(a + b)$,
- la soustraction $(a - b)$,
- la multiplication ($a \times b$ ou ab),
- la division euclidienne (que nous allons définir précisement dans ce cours).

2.1) divisibilité, nombres premiers et division euclidienne:

Définition 1:
$\forall\, (a, b) \in \mathbb{Z}^2 \wedge b \neq 0,\ b|a \Leftrightarrow (\exists q \in \mathbb{Z}, a = bq)$.
la notation $b|a$ se lit "b divise a " ou "a est multiple de b".

Remarques, vocabulaire :

- Si $b|a$ et si $b > 0$ on dit que b est un diviseur de a . Le mot "diviseur" est donc réservé aux entiers strictement positifs.
- Si $b|a$ et si $a \neq 0$ alors $|b| \leq |a|$
- On remarquera que si $b|a$ alors $-b|a$.
- Les diviseurs de 28 sont $1, 2, 4, 7, 14$ et 28. Ceux sont aussi les diviseurs de -28.

Propriétés 1:
$\forall\, (a, b, c) \in \mathbb{Z}^3$:

1. $(c|a) \wedge (c|b) \Rightarrow \forall\, (u, v) \in \mathbb{Z}^2, (c|au + bv)$ (en particulier c divise $a + b$ et $a - b$)
2. $(c|b) \wedge (b|a) \Rightarrow c|a$
3. $(a|b) \wedge (b|a) \Rightarrow a = \pm b$
4. Si $a \neq 0$ alors $(-1|a) \wedge (1|a) \wedge (a|a) \wedge (-a|a)$ (donc tout nombre entier strictement positif, sauf éventuellement 1, a au moins deux diviseurs 1 et lui-même)
5. Si $b \neq 0$ alors $b|0$ (attention: la notation $b|a$ n'a été définie que pour $b \neq 0$. Finalement 0 a une infinité de diviseurs et 1 n'en a qu'un seul. Attention: -1 divise 1 mais n'est pas un diviseur de 1)

(la démonstration de ces 5 propriétés est simple à partir de la définition 1 et peut être laissée en exercice).

Définition 2:
$p \in \mathbb{N}^*$ est un **nombre premier** s'il n'a que 2 diviseurs: 1 et lui-même.

(donc 1 n'est pas un nombre premier).

Liste des nombres premiers ≤ 100 : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97 .

Propriété 2:
Un entier $n \geq 3$ qui n'est divisible par aucun des nombres premiers compris entre 2 et $\sqrt{n}$ est premier.

(démonstration par l'absurde)

Propriété 3 :

Il existe une infinité de nombres premiers.

Démonstration par l'absurde :
Supposons qu'il n'existe que k nombres premiers notés $p_1, p_2, \ldots, p_k$. Soit $n = p_1 p_2 \ldots p_k + 1$ et p un facteur premier de n. p est différent de tous les p_i (sinon p diviserait $n - p_1 p_2 \ldots p_k = 1$) ce qui est absurde.

définition 3 (et propriété) : la division euclidienne
Soient a et b deux entiers, si $b \neq 0$ alors il existe 2 entiers uniques q et r vérifiant $a = bq + r$ et $0 \leq r < |b|$.
Le calcul de q et de r s'appelle la division euclidienne de a par b. q est le quotient, r est le reste.

Démonstration de l'existence et de l'unicité de q et r :

1. **démonstration de l'unicité par l'absurde** : Supposons l'existence de deux couples de solutions (q, r) et (q', r'). Par définition on a $a = bq + r$ et $a = bq' + r'$ avec $0 \leq r < |b|$ et $0 \leq r' < |b|$.
   Les deux premières égalités donnent $bq + r = bq' + r'$ c'est-à-dire $b(q - q') = (r' - r)$ d'où on en déduit que $(r' - r)$ est un multiple de b. Or, d'après les deux inégalités,$|r' - r| < |b|$, donc $r' - r$ ne peut pas être un multiple de b différent de 0, d'où on en déduit que $r = r'$ et $q = q'$, d'où l'unicité du couple de solution.
2. **Existence de q et r :**
   supposons, sans perte de généralité, que $b > 0$.
   On considère l'ensemble $E = \{a - kb, k \in \mathbb{Z} \wedge a - kb \geq 0\}$.
   E est non vide car $a \in E$ si $a > 0$, sinon $a - ba = -a(b - 1) \in E$.
   Soit r le plus petit élément de E (On admettra que dans $\mathbb{N}$, tout sous-ensemble non vide admet un plus petit élément pour la relation d'ordre $\leq$).
   Alors, par définition de l'ensemble E, il existe un entier k tel que $r = a - kb$ soit $a = bk + r$. On a, par définition de r, $r \geq 0$;
   il reste donc à démontrer que $r < b$:
   Supposons (par l'absurde) que $r \geq b$:
   alors $r > r - b \geq 0$, or $r - b = a - kb - b = a - (k + 1)b \in E$. Ce qui est absurde puisque r est le plus petit élément de E (et on vient d'en trouver un encore plus petit). Donc, finalement, $r < b$.

Exemples : (attention aux signes !)

- division euclidienne de 16 par 3 : 16 = 3 x 5 + 1 donc q= 5 et r = 1
- division euclidienne de 16 par -3 : 16 = (-3) x (-5) + 1 donc q= -5 et r = 1
- division euclidienne de -16 par 3 : -16 = 3 x (-6) + 2 donc q= -6 et r = 2
- division euclidienne de -16 par -3 : -16 = (-3) x 6 + 2 donc q = 6 et r = 2

2.2) pgcd, ppcm, Bézout:

Définition 4 (et théorème) :

Soient a et b deux éléments de $\mathbb{Z}$ avec $(a, b) \neq (0, 0)$. Si d est un diviseur de a et de b à la fois alors $1 \leq d \leq \inf(|a|, |b|)$. L'ensemble des diviseurs de a et de b est donc fini, il possède donc un plus grand élément noté $p\gcd(a, b)$ ou $a \wedge b$ (« plus grand commun diviseur »).

Définition 5:

Soient a et b deux éléments de $\mathbb{Z}$ avec $(a, b) \neq (0, 0)$.
On appelle "plus petit commun multiple" de a et b, et on note $\text{ppcm}(a, b)$ ou $a \vee b$, le plus petit entier positif qui soit multiple de a et de b.

exemples:
$12 \wedge 30 = 6$; $27 \wedge 0 = 27$.

$12 \vee 30 = 60; 27 \vee 0 = 0$.

Propriétés 4 (propriété immédiates du PGCD de deux entiers):

Soient a et b deux éléments de $\mathbb{Z}$ avec $(a, b) \neq (0, 0)$.

1. $a \wedge b = b \wedge a$
2. $a \wedge b = |a| \wedge |b|$
3. $a \wedge 1 = 1$
4. $a \wedge 0 = |a|$
5. $a \wedge a = |a|$
6. $a \wedge b = b \Leftrightarrow b|a$
7. si p est premier alors $a \wedge p = p$ si p divise a, sinon $a \wedge p = 1$.
8. si r est le reste de la division euclidienne de a par b alors $a \wedge b = b \wedge r$

Démonstration du point 8:
Par définition de la division euclidienne $a = bq + r$ et $r = a - bq$.
Donc, d'après la propriété 1 point 1 et la première égalité ci-dessus, un diviseur d de b et r est aussi un diviseur de a et b.
De même un diviseur d de a et b est aussi un diviseur de b et r d'après la deuxième égalité.
Donc les diviseurs communs à a et b sont les mêmes que ceux de b et r , d'où la conclusion.

Ce point 8 de la propriété 4 permet de justifier l'algorithme récursif ci-dessous (algorithme d'Euclide) permettant de calculer le pgcd de deux nombres entiers a et b:

Euclide(a, b)
si $b == 0$
retourner a
sinon
retourner **Euclide**(b, r)

Exemple:

Euclide(70,28)=Euclide(28,14)=Euclide(14,0)=14

Coût de l'algorithme d'Euclide:

On peut démontrer que le nombre d'appels récursifs de la fonction Euclide(a, b) dans le cas où $a > b \geq 0$ est en $O(\lg(b))$ c'est à dire en $O(\beta)$ si β est le nombre de bits nécessaires à l'encodage de b .
Si l'on admet que la multiplication et la division coûtent $O\left(\beta^2\right)$, le coût de l'algorithme Euclide en opérations de bits est $O\left(\beta^3\right)$.
Le coût de l'algorithme d'Euclide est donc polynomial en fonction du nombre de bits de l'entrée.

Définition 6:
Deux entiers a e b sont premiers entre eux $\Leftrightarrow a \wedge b = 1$

Propriété 5: Bézout

Soient a et b deux entiers fixés. Alors l'ensemble des nombres de la forme $au + bv$ où u et v sont des entiers relatifs, coïncide avec l'ensemble des multiples de $a \wedge b$.

Démonstration :
Soient $E = \left\{au + bv, (u, v) \in \mathbb{Z}^2\right\}$ et $F = \{x \in \mathbb{Z}, (a \wedge b) | x\}$.
Démontrons que $E = F$:

On a $E \subseteq F$ car $a \wedge b$ divise a et b par définition, donc $a \wedge b$ divise $au + bv$ quelques soient u et v entiers.

Il nous reste à monter que $F \subseteq E$:

Soit d le plus petit élément strictement positif de E . Soit x un élément quelconque de E.

La division euclidienne de x par d donne $x = dq + r$ avec $0 \leq r < d$. On peut aussi écrire $r = x - dq$ et donc r est un élément de E (car on montre facilement que r est une combinaison linéaire de a et de b puisque x et d sont des combinaisons linéaires de a et b).

Mais puisque d est le plus petit élément strictement positif de E, on a forcément $r = 0$. On vient donc de démontrer que tous les éléments de E sont des multiples de d. Inversement, puisque E est stable pour la multiplication, tous les multiples de d sont dans E.

Donc finalement $E =$\{ les multiples de d \}. Démontrons enfin que $d = a \wedge b$:

On a déjà démontré que $a \wedge b$ divise tous les éléments de E, donc $a \wedge b$ divise $d \neq 0$ et par conséquent $a \wedge b \leq d$. De plus d divise a et b (car a et b sont dans E) donc $d \leq a \wedge b$ (car $a \wedge b$ est le plus grand diviseur commun à a et b).Finalement $a \wedge b = d$.

On vient donc de démontrer que **l'équation** $au + bv = a \wedge b$ **admet au moins une solution**
(puisque d est un élément de E et que $d = a \wedge b$).
On appelle ce résultat **l'identité de Bezout.**

Dans le cas particulier de 2 nombres premiers entre eux ($a \wedge b = 1$), on a l'existence de u et v entiers tels que $au + bv = 1$

Propriétés 6:
$\forall a, b, c, d \in \mathbb{Z}$:

1. $d|a \wedge d|b \Leftrightarrow d|\,(a \wedge b)$
2. $(ca) \wedge (cb) = c\,(a \wedge b)$
3. $c|a$ et $c|b \Rightarrow (a/c) \wedge (b/c) = (a \wedge b)\,/c$
4. $(a/d) \wedge (b/d) = 1 \Leftrightarrow d = a \wedge b$
5. $a \wedge b = 1$ et $a|bc \Rightarrow a|c$ (c'est le **lemme de Gauss**)
6. $a \wedge b = 1$ et $a \wedge c = 1 \Leftrightarrow a \wedge bc = 1$
7. $a \wedge b = 1 \Rightarrow a \wedge bc = a \wedge c$
8. p est premier et $p|ab \Rightarrow p|a$ ou $p|b$ (c'est le **lemme d'Euclide**)

Attention: dans l'écriture des propriétés précédentes il ne faut pas confondre les symboles | et / ni confondre le "et" logique qu'on peut représenter par le symbole $\wedge$ tout comme le pgcd de deux entiers.

Démonstration:

point 1:
si d divise a et d divise b alors d divise toute combinaison linéaire de a et b, en particulier $a \wedge b$ (qui est une combinaison linéaire de a et b d'après l'identité de Bézout).

La réciproque se démontre facilement puisque par définition $(a \wedge b)\,|a$ et $(a \wedge b)\,|b$. Donc si $d|\,(a \wedge b)$ la conclusion découle du point 2 des propriétés 1.

point 5, lemme de Gauss:
$a \wedge b = 1$ donc d'après l'identité de Bézout il existe u et v deux entiers tels que $au + bv = 1$.

En multipliant par c cette égalité on obtient $acu + bcv = c$. Donc c est une combinaison linéaire de a et de bc d'où on déduit que $a|c$

Point 8, lemme d'Euclide:
Soit p un nombre premier tel que p/ab . Si p ne divise pas a alors $p \wedge a = 1$ (car p est premier) et d'après le lemme de Gauss on a p/b

Les autres démonstrations peuvent être laissées en exercices.

2.3) Algorithme d'Euclide étendu:(cf livre Algorithmique, Cormen, Leiserson, Rivest, Stein)

Cet algorithme permet de calculer simultanément $a \wedge b$ et deux entiers x et y tels que $ax + by = a \wedge b$.
Il prend en entrée deux entiers positifs:

Euclide-Etendu(a, b)

1. **Si** $b == 0$ alors
2. **retourner** $(a, 1, 0)$
3. **sinon** $(d', x', y') =$ **Euclide-Etendu**(b, r) // r est le reste dans la division euclidienne de a par b
   (a) $(d, x, y) = (d', y', x' - qy')$ // q est le quotient de la division euclidienne de a par b
   (b) **retourner** (d, x, y)

Exemple de trace de l'algorithme:

entrées			sorties		
a	b	q	d	x	y
47	25	1	1	8	-15
25	22	1	1	-7	8
22	3	7	1	1	-7
3	1	3	1	0	1
1	0		1	1	0

Dans cette trace, on calcule les entrées successives (a, b) de la fonction Euclide-Etendu du haut vers le bas (on peut aussi calculer q) puis les sorties se font du bas vers le haut.

Démonstration de l'algorithme:

Nous allons démontrer l'algorithme **Euclide-Etendu(a,b)** par une récurrence seconde forme sur l'entier b :
$\forall b \in \mathbb{N}$: "Pour tout entier $a \in \mathbb{N}$ l'algorithme **Euclide-Etendu(a,b)** renvoie un triplet (d, x, y) tel que $d = a \wedge b$ et $ax + by = d$."
On note $P(b)$ le prédicat entre guillemets ci-dessus. On a déjà vu avec l'algorithme **Euclide(a,b)** que $d = a \wedge b$ mais on va tout redémontrer ici:

$P(0)$?**(on cherche ici à démontrer que le prédicat est vrai pour $b = 0$):**
Dans ce cas, les lignes 1 et 2 de l'algorithme nous assurent que le triplet renvoyé est $(a, 1, 0)$ qui vérifie parfaitement les conditions requises. Donc $P(0)$ est vraie.

Supposons que le prédicat est vrai pour tous les entiers strictement inférieurs à $b > 0$, démontrons alors qu'il est vrai pour b :

D'après la ligne 3a de l'algo on a $E0 : (d, x, y) = (d', y', x' - qy')$ avec, par hypothèse de récurrence (d'après la ligne 3 et puisque $0 \leq r < b$) :
$E1 : d' = b \wedge r$ et $E2 : bx' + ry' = d'$.
De plus par définition $E3 : a = qb + r$ (dans l'algo q et r sont le quotient et le reste de la division euclidienne de a par b).
D'après $E0$ et $E1$ on a donc $d = d' = b \wedge r$ et d'après la propriété 4.8 de ce cours $b \wedge r = a \wedge b$ ce qui démontre la première partie du prédicat.

Il nous reste a démontrer que $ax + by = d$ (on va partir du premier membre):

$ax + by = ay' + b(x' - qy')$ d'après $E0$.

$\ldots = (qb + r)y' + b(x' - qy') = bx' + ry' = d' = d$ (on a remplacé a par $bq + r$ d'après $E3$, puis on a développé et simplifié, et on a utilisé $E2$). On obtient le résultat voulu donc l'algorithme est démontré.

2.3) décomposition d'un nombre premier en un produit de facteurs premier:

Propriété 7 :
Tout entier $n \geq 2$ est soit premier, soit un produit de nombres premiers. La décomposition d'un nombre en un produit de facteurs premiers est unique (si on ne tient pas compte de l'ordre des facteurs).

Plus formellement l'entier $n \in \mathbb{N}^*$ peut s'écrire:

$$n = p_1^{k_1} p_2^{k_2} \dots p_s^{k_s}$$

où les entiers naturels p_i sont premiers et $p_1 < p_2 < \ldots < p_s$ et les k_i sont des entiers strictement positifs.

démonstration:

1. existence par récurrence seconde forme sur n:
   (a) La propriété est vraie pour $n = 2$.
   (b) On suppose la propriété démontrée pour tout entier inférieur ou égal à n. Considérons $n+1$: soit $n+1$ est premier et la propriété est établie, soit $n+1$ admet un diviseur premier q et donc $n+1 = pq$ avec $p \leq n$. On conclut en utilisant l'hypothèse de récurrence sur p .

2. Unicité de la décomposition: On raisonne par l'absurde et on utilise le lemme d'Euclide.
   Soient deux factorisations f_1 et f_2 d'un même nombre n et soit p un facteur premier apparaissant dans la première factorisation. Il divise donc la deuxième factorisation (puisqu'elles sont égales à n toutes les deux). Si p était différent de tous les facteurs premiers de f_2 (donc premier avec ces facteurs) il ne pourrait pas diviser le produit de ces facteurs, c'est à dire f_2 (c'est l'application de la contraposée du lemme d'Euclide). p apparaît donc dans f_2 . On simplifie f_1 et f_2 par p et on recommence le même raisonnement sur les nouvelles factorisations (c'est une récurrence). On obtient finalement que les deux factorisations sont identiques.

Méthode pour décomposer un nombre n en facteurs premiers :

1. Déterminer le plus petit diviseur premier de n autre que 1.
2. Diviser n par ce facteur premier . On note m le quotient.
3. Si $m > 1$ faire $n = m$ et recommencer au 1).

Exemple:

5888	2
2944	2
1472	2
736	2
368	2
184	2
92	2
46	2
23	23
1	

Donc $5888 = 2^8 \times 23$

Coût de l'algorithme de décomposition dans un cas particulier:

Nous allons estimer le coût de calcul dans le cas particulier où $n = p_1 p_2$ avec n un entier codé sur β bits et p_1 et p_2 deux nombres premiers codés chacun sur $\frac{\beta}{2}$ bits ($p_1 < p_2$):

On admettra qu'il n'existe pas de formule explicite donnant la liste des nombres premiers $\leq \sqrt{n}$. C'est donc l'étape 1 de l'algorithme de décomposition qui sera la plus longue. On peut procéder naïvement en essayant tous les entiers impairs (et aussi 2) inférieurs ou égaux à p_1. Cet algorithme serait en $\Omega(p_1)$ et donc finalement en $\Omega\left(2^{\frac{\beta}{2}}\right)$. Pour des grands nombres entiers (avec $\frac{\beta}{2} > 500$) un tel algorithme est irréalisable. Il n'existe pas d'algorithme efficace pour réaliser cette décomposition.

III) L'anneau $\mathbb{Z}/n\mathbb{Z}$ (arithmétique modulaire)

3.1)Définition des congruences dans $\mathbb{Z}$:

définition 7:
Soit un entier $n > 1$. On dit que 2 entiers sont congrus modulo n si $a - b$ est un multiple de n .
On note $a \equiv b \bmod n$ ou plus simplement $a \equiv b\,(n)$

Remarques:

On a donc $a - b = kn$ (avec $k \in \mathbb{Z}$) et a et b ont le même reste dans la division euclidienne par n .

Exemples:

on a $153 \equiv 20\,(7)$ car $153 - 20 = 19 \times 7$.
De plus 153 et 20 ont le même reste dans la division euclidienne par 7 :
$153 = 21 \times 7 + 6$ et $20 = 2 \times 7 + 6$.
On peut donc écrire aussi que $153 \equiv 6\,(7)$.

Proposition 8: compatibilité de l'addition et de la multiplication

Si $a \equiv a'\,(n)$ et $b \equiv b'\,(n)$ alors:

- $a + b \equiv a' + b'\,(n)$
- $a \times b \equiv a' \times b'\,(n)$
- $a^k \equiv a'^k\,(n)\,, \forall k \in \mathbb{N}$

Démonstrations: elles peuvent se faire en exercice

Exemples:

- $7x + 53 \equiv x + 5\,(6)$
- $129^{124} \equiv (-1)^{124}\,(13)$ car $129 \equiv -1\,(13)$. On a donc finalement $129^{124} \equiv 1\,(13)$

Proposition 9:

Pour un entier $n > 1$ fixé, la relation binaire $a \equiv b\,(n)$ sur $\mathbb{Z}$ est une relation d'équivalence ce qui signifie qu'elle est:

- réflexive: $\forall a \in \mathbb{Z}, a \equiv a\,(n)$
- symétrique: $\forall\,(a,b) \in \mathbb{Z}^2, a \equiv b\,(n) \Rightarrow b \equiv a\,(n)$
- transitive: $\forall\,(a,b,c) \in \mathbb{Z}^3, a \equiv b\,(n) \wedge b \equiv c\,(n) \Rightarrow a \equiv c\,(n)$

Définition 8 (et propositions):

- Dans une relation d'équivalence R sur un ensemble E , on appelle classe d'équivalence de l'élément $x \in E$, l'ensemble des éléments de E en relation avec x . On note $\dot{x}$ cette classe d'équivalence.
- Deux éléments x et y de E tels que xRy ont la même classe d'équivalence.
- Les classes d'équivalence forment une partition de l'ensemble E .

Application de cette définition à la relation de congruence modulo n :

Considérons pour commencer le cas où $n = 3$:
On a:

- $\overset{\bullet}{0} = \{3k, k \in \mathbb{Z}\} = \overset{\bullet}{3} = \overset{\bullet}{6} = \overset{\bullet}{-3} = \ldots$
- $\overset{\bullet}{1} = \{3k+1, k \in \mathbb{Z}\} = \overset{\bullet}{4} = \overset{\bullet}{7} = \overset{\bullet}{-2} = \ldots$
- $\overset{\bullet}{2} = \{3k+2, k \in \mathbb{Z}\} = \overset{\bullet}{5} = \overset{\bullet}{8} = \overset{\bullet}{-1} = \ldots$

Il n'y a donc que 3 classes d'équivalence distinctes. On notera $\mathbb{Z}/3\mathbb{Z} = \left\{\overset{\bullet}{0}, \overset{\bullet}{1}, \overset{\bullet}{2}\right\}$.

Attention: une classe d'équivalence est un ensemble d'entiers. La classe $\overset{\bullet}{0}$ contient les entiers multiples de 3, on a donc $\overset{\bullet}{0} = \overset{\bullet}{3}$ (par exemple). Une classe d'équivalence admet donc une infinité de représentants (i.e. d'éléments). On choisira de préférence le plus petit représentant positif.

L'ensemble $\mathbb{Z}/3\mathbb{Z} = \left\{\overset{\bullet}{0}, \overset{\bullet}{1}, \overset{\bullet}{2}\right\}$ peut aussi s'écrire $\mathbb{Z}/3\mathbb{Z} = \{0, 1, 2 \bmod 3\}$ en convenant que la notation $x \bmod 3$ remplace $\overset{\bullet}{x}$ et en n'écrivant qu'une seule fois $\bmod 3$ pour simplifier.

Dans le cas général d'un entier $n > 1$ quelconque, la relation de congruence modulo n admet exactement n classes d'équivalences distinctes: $\mathbb{Z}/n\mathbb{Z} = \{0, 1, 2, \ldots, (n-1) \bmod n\}$.

Définition 9: Addition et multiplication dans $\mathbb{Z}/n\mathbb{Z}$.

- $(a \bmod n) + (b \bmod n) = (a+b) \bmod n$
- $(a \bmod n)(b \bmod n) = (ab) \bmod n$

En général pour simplifier on n'écrira qu'une seule fois en fin de ligne l'expression $mod\, n$.

En utilisant la proposition 8 sur la compatibilité de l'addition et de la multiplication avec la congruence modulo n , on peut montrer que l'addition et la multiplication définis comme ci-dessus ont un sens (en particulier le résultat de ces opérations ne dépend pas du représentant choisi):

$2 + 1 = 0 \bmod 3$ mais on a aussi $5 - 2 = 3 \bmod 3$. Ces deux égalités représentent le même calcul dans $\mathbb{Z}/3\mathbb{Z}$ car 2 et 5 sont dans la même classe (donc $2 = 5 \bmod 3$), ainsi que 1 et -2 et aussi 0 et 3.

Les règles de calculs usuels dans $\mathbb{Z}$ pour les lois internes $+, -, \times$ (addition, soustraction et multiplication) restent valables dans $\mathbb{Z}/n\mathbb{Z}$**:**

- $x + 0 = x \bmod n$
- $x + (-x) = 0 \bmod n$
- $x \times 1 = x \bmod n$
- $x \times 0 = 0 \bmod n$
- $x(y+z) = xy + xz \bmod n$
- $(x \bmod n)^k = x^k \bmod n$...

Attention à la division: $2x \equiv 4\,(20)$ ne se simplifie pas en $x \equiv 2\,(20)$

3.2) Algorithme d'exponentiation-modulaire:

Dans les chiffrements asymétriques que nous verrons (RSA et El Gamal) l'une des opérations principales est le calcul d'une puissance d'un nombre entier modulo n : $a^b \bmod n$. Pour 3 entiers a, b et n encodés chacun sur β bits on pourrait effectuer $b-1$ multiplications consécutives ce qui donnerait un algorithme en $\Omega\left(\beta^2 \times 2^\beta\right)$ donc non polynomial. L'algorithme suivant permet de réaliser ce calcul en $O\left(\beta^3\right)$:

Exponentiation-modulaire(a, b, n)

1. $d \leftarrow 1$
2. soit $b_{\beta-1}b_{\beta-2}\dots b_1b_0$ la représentation de b en base 2
3. **Pour** $i = \beta - 1$ **jusqu'à** 0 **en décrémentant de** -1 **faire:**
   (a) $d \leftarrow (d \times d) \bmod n$
   (b) **Si** $b_i == 1$ **faire:**
      i. $d \leftarrow (d \times a) \bmod n$
   (c) **Fin Si**
4. **Fin Pour**
5. retourne d

Exemple de trace de l'algorithme d'exponentiation modulaire:

On veut calculer $7^{560}\,(561)$. On a donc $b = 560 = 1000110000_2$, $\beta = 10$ (nombre de bits), $a = 7$ et $n = 561$:

b_i	1	0	0	0	1	1	0	0	0	0
d_i	7	49	157	526	160	241	298	166	67	1

donc $7^{560} = 1\,(561)$

Démonstration de l'algorithme:
On peut faire une démonstration par récurrence sur $\beta \geq 1$ le nombre de bits de b:
$\forall \beta \geq 1$: "Pour tous les entiers a, n et pour tout entier b dont l'écriture en base 2 est de longueur β l'algorithme **Exponentiation-modulaire**(a, b, n) renvoie $a^b \bmod n$ "

Notons $P(\beta)$ le prédicat entre guillemets.

$P(1)$?
Dans ce cas l'entier b admet une écriture en base 2 de la forme b_0 avec $b_0 = 0$ ou $b_0 = 1$.
Dans l'algorithme on initialise la valeur de d à 1 (ligne 1) et on ne passe qu'une seule fois dans la boucle.
On calcule $d \leftarrow (d \times d) \equiv 1 \bmod n$ et on s'arrête là si $b_0 = 0$ sinon on calcule $d \leftarrow (d \times a) \equiv a \bmod n$. Dans les deux cas la valeur renvoyée est bien $a^b \bmod n$

Démontrons l'implication $P(\beta - 1) \Rightarrow P(\beta)$ pour tout entier $\beta \geq 2$:
D'après l'hypothèse de récurrence, le passage dans la boucle $\beta - 1$ fois permet de calculer $d \leftarrow a^{\frac{b-b_0}{2}}\,(n)$ (car $\frac{b-b_0}{2} = b_{\beta-1}b_{\beta-2}\dots b_1$ est un entier encodé sur $\beta - 1$ bits).
En passant une dernière fois dans la boucle on calcule $d \leftarrow d^2 \equiv a^{b-b_0}$ puis si $b_0 = 1$ on calcule $d \leftarrow d \times a \equiv a^b\,(n)$. Dans les deux cas on renvoie $a^b \bmod n$

3.3) Eléments inversibles de $\mathbb{Z}/n\mathbb{Z}$ et fonction d'Euler:

Définition 10:

Soit a un entier, on dit que a est inversible modulo n s'il existe b tel que $ab \equiv 1 \bmod n$. On note $a^{-1} \bmod n$ cet inverse.
L'ensemble des éléments inversible de $\mathbb{Z}/n\mathbb{Z}$ est noté $(\mathbb{Z}/n\mathbb{Z})^*$.

Exemples:

- $(\mathbb{Z}/4\mathbb{Z})^* = \{1, 3 \bmod 4\}$.
- $(\mathbb{Z}/6\mathbb{Z})^* = \{1, 5 \bmod 6\}$.

Proposition 10:

$(\mathbb{Z}/n\mathbb{Z})^* = \{a \, mod \, n, (1 \leq a \leq n-1) \text{ et } (a \wedge n = 1)\}$
Pour déterminer l'inverse d'un élément a modulo n il suffit de déterminer u dans la relation de Bézout $au + nv = 1$.
Alors $a^{-1} = u \bmod n$

Démonstration:

- Si $ab \equiv 1 \bmod n$ alors par définition $ab = 1 + qn$ donc $ab - qn = 1$ d'où on déduit que $a \wedge n = 1$.
- inversement si $au + nv = 1$ on en déduit que $au = 1 - nv$ donc a est inversible modulo n .

Le calcul de l'inverse se fait donc grâce à l'algorithme Euclide-etendu.

Conséquence:

Si p est premier alors tout élément non nul de $\mathbb{Z}/p\mathbb{Z}$ est inversible modulo p .

Définition 11 (fonction d'Euler):

La fonction φ d'Euler est définie, pour tout $n \geq 1$ entier par:

$$\varphi(n) = \left|(\mathbb{Z}/n\mathbb{Z})^*\right|$$

(c'est donc le nombre de nombres premiers avec n, inférieurs ou égaux à n).

Proposition 11:

1. $\varphi(1) = 1$
2. Pour tout nombre premier p on a $\varphi(p) = p - 1$
3. Pour tout nombre premier p et tout entier $k \geq 1$ on a $\varphi\left(p^k\right) = p^k - p^{k-1}$
4. Pour tout m, n tels que $m \wedge n = 1$ on a $\varphi(mn) = \varphi(m)\varphi(n)$

démonstration:

Les points 1 et 2 sont évidents.

Point 3:
Les entiers qui ne sont pas premiers avec p^k sont de la forme pq avec $q \in \left\{1, 2, \ldots, p^{k-1}\right\}$. Il y en a donc p^{k-1} d'où $\varphi\left(p^k\right) = p^k - p^{k-1}$.
Par contre la démonstration du point 4 se fait à partir du théorème "chinois" que je ne montre pas dans ce cours. Ce dernier point sera important pour l'algorithme RSA.

3.4) **Le théorème d'Euler et le petit théorème de Fermat:**

Proposition 12 (Théorème d'Euler):

Pour tous les entiers $n \geq 1$ et a tels que $a \wedge n = 1$, on a $a^{\varphi(n)} \equiv 1\,(n)$

Démonstration:

Puisque a est inversible dans $\mathbb{Z}/n\mathbb{Z}$ (car $a \wedge n = 1$) l'application $x \to ax$ est une bijection de $(\mathbb{Z}/n\mathbb{Z})^*$ dans $(\mathbb{Z}/n\mathbb{Z})^*$. En effet tout élément de $(\mathbb{Z}/n\mathbb{Z})^*$ admet un antécédent unique par l'application réciproque $x \to a^{-1}x$.
On a donc l'égalité ensembliste suivante: $(\mathbb{Z}/n\mathbb{Z})^* = \left\{ax, x \in (\mathbb{Z}/n\mathbb{Z})^*\right\}$.
D'où on déduit l'égalité dans $\mathbb{Z}/n\mathbb{Z}$ suivante: $\prod_{x \in (\mathbb{Z}/n\mathbb{Z})^*} x = \prod_{x \in (\mathbb{Z}/n\mathbb{Z})^*} ax = a^{\varphi(n)} \prod_{x \in (\mathbb{Z}/n\mathbb{Z})^*} x \bmod n$
Si on pose $P = \prod_{x \in (\mathbb{Z}/n\mathbb{Z})^*} x \bmod n$ on a donc $P = a^{\varphi(n)} P \bmod n$ donc $a^{\varphi(n)} \equiv 1 \bmod n$ car P est inversible puisque c'est le produit de tous les éléments inversible de $(\mathbb{Z}/n\mathbb{Z})^*$.

Conséquence (le petit théorème de Fermat):

Pour tout entier p premier et pour tout entier a on a $a^p \equiv a\,(p)$

l'ensemble $(\mathbb{Z}/n\mathbb{Z})^*$ est muni d'une loi interne associative (la multiplication), d'un élément neutre ($1 \bmod n$) et chaque élément possède un inverse: une telle structure mathématique s'appelle **un groupe.**

Proposition 13:

Soit $x \in (\mathbb{Z}/n\mathbb{Z})^*$. La suite $\left(x^k\right)_{k\in\mathbb{Z}}$ est périodique. Notons $\omega(x)$ sa période (c'est à dire le pus petit entier strictement positif tel que pour tout $k \in \mathbb{Z}$ on ait $x^{k+\omega(x)} \equiv x^k\,(n)$
Alors $x^l \equiv 1\,(n) \Leftrightarrow \omega(x)\,|l$

Conséquence: on a donc en particulier $\omega(x)\,|\varphi(n)$.

Notation:

On notera $< x >= \left\{1, x, x^2, \ldots, x^{\omega(x)-1} \bmod n\right\}$ le sous-ensemble de $(\mathbb{Z}/n\mathbb{Z})^*$engendré par x .

Démonstration de la proposition 13:
Il faut avant tout démontrer la périodicité de la suite $\left(x^k\right)_{k\in\mathbb{Z}}$. Or cette suite prend ses valeurs dans un ensemble fini donc il existe nécessairement deux entiers distincts $i < j$ tels que $x^i = x^j \bmod n$ et puisque x^i est inversible on a $x^{j-i} \equiv 1 \bmod n$ donc la suite est $j - i$ périodique (attention dans cette démonstration $j - i$ n'est pas nécessairement égal à $\omega(x)$. Déterminer $\omega(x)$ est en général un problème difficile). Le reste de la démonstration est assez simple.

Exemple:

Dans $(\mathbb{Z}/7\mathbb{Z})^*$ on a :

- $< 2 >= \{1, 2, 4 \bmod n\}$ et donc $\omega(2) = 3$. On constate que $\omega(2)$ divise $\varphi(7)$.
- $< 3 >= (\mathbb{Z}/7\mathbb{Z})^*$et donc $\omega(3) = 6 = \varphi(7)$.
- $< 6 >= \{1, 6 \bmod n\}$

(à vérifier).

Proposition 14:

Pour tout entier p premier il existe $x \in (\mathbb{Z}/p\mathbb{Z})^*$ tel que $< x >= (\mathbb{Z}/p\mathbb{Z})^*$ (donc $\omega(x) = \varphi(p) = p - 1$).
On dit que $(\mathbb{Z}/p\mathbb{Z})^*$ est un groupe cyclique.

Exemples:

- dans l'exemple précédent on a vu que $< 3 >= (\mathbb{Z}/7\mathbb{Z})^*$. Donc 3 est un générateur de $(\mathbb{Z}/7\mathbb{Z})^*$. Ce n'est pas le cas de 2 .
- 2 est un générateur de $(\mathbb{Z}/11\mathbb{Z})^*$ (à vérifier).

3.5) Logarithme discret:

définition 12:
Calculer le logarithme discret base a c'est, étant donné $A = a^x \bmod n$, déterminer x dans $\mathbb{Z}/n\mathbb{Z}$.
Ce calcul n'est possible que si $x \longmapsto a^x \bmod n$ est une bijection. On notera $x = \log_a(A)$

Exemples:

Dans $\mathbb{Z}/11\mathbb{Z}$ on a $\log_2(5) = 4 \bmod 11$ car $2^4 \equiv 5\,(11)$.
Par contre dans $\mathbb{Z}/7\mathbb{Z}$, $\log_2(5)$ n'existe pas.

Si on choisit bien l'entier n et l'entier a le problème du logarithme discret est un problème difficile à résoudre algorithmiquement (i.e non polynomial). C'est sur cette impossibilité qu'est basée le chiffrement d'El Gamal.

IV) Cryptographie asymétrique:

4.1)Introduction:

4.2) RSA:

(Ronald Rivest, Adi Shamir, Leonard Adleman)

définition d'une clé RSA:

Une clé RSA est un couple $k = (k_{pub}, k_{priv})$ définie de la manière suivante:

1. on se donne p et q deux "grands" nombres premiers distincts et de même taille binaire.
2. on se donne e et d deux entiers tels que $ed \equiv 1 mod\left((p-1)(q-1)\right)$
   e est appelé "exposant de chiffrement", d est appelé "exposant de déchiffrement".
3. on calcule $N = pq$ (N est appelé module de chiffrement)

Alors $k_{pub} = (N, e)$ et $k_{priv} = d$ ou $k_{priv} = (N, d)$(on parle de clé publique et de clé privée).

Utilisation de la clé RSA pour chiffrer un message m :

A veut envoyer un message m (on note m le message avant le chiffrement) à B de façon à ce que B soit le seul à pouvoir le déchiffrer. La clé publique $k_{pub} = (N, e)$ est connue de tous les utilisateurs du réseau, seule la clé privée $k_{priv} = d$ n'est connue que de B. C'est donc B qui a créé cette clé et qui a transmis la clé publique à toutes les personnes susceptibles de communiquer avec lui.

Le message m sera un élément de $\mathbb{Z}/N\mathbb{Z}$ (donc un nombre entier $\in \{0, 1, \ldots, N-1\}$).

On notera $c = E(m)$ le message chiffré (E est une application de $\mathbb{Z}/N\mathbb{Z}$ dans $\mathbb{Z}/N\mathbb{Z}$).

La fonction de déchiffrement sera noté D (D est aussi une application de $\mathbb{Z}/N\mathbb{Z}$ dans $\mathbb{Z}/N\mathbb{Z}$). On doit donc avoir $D(E(m)) = m$ pour tout message en clair m . Avec la clé RSA k les fonctions E et D sont définies de la façon suivante:

- $E(m) = m^e mod(N)$
- $D(c) = c^d mod(N)$

Vérifions que $D(E(m)) = m$ pour tout message en clair m :

1er cas:$m \wedge N = 1$:

$D(E(m)) = (m^e)^d = m^{ed} = m^{1+k(p-1)(q-1)} = m \times \left(m^{\varphi(N)}\right)^k \equiv m\, mod(N)$ (car $m^{\varphi(N)} \equiv 1 mod(N)$ d'après le théorème d'Euler)

2ème cas: $m \wedge N \neq 1$ donc m est multiple de p ou m est multiple de q (Si $m \neq 0$ alors m ne peut pas être multiple de N car $0 < m < N$. Le cas $m = 0$ est évident).

Posons $t = 1 + k(p-1)(q-1)$

Supposons par exemple que m est multiple de p donc $m \wedge q = 1$ et par conséquent (Th d'Euler) $m^{q-1} \equiv 1 mod(q)$ d'où on en déduit que $m^{(q-1)(p-1)} \equiv 1 mod(q)$ et enfin que $m^t \equiv m mod(q)$.

De plus, puisque m est multiple de p on a $m^t \equiv m \equiv 0 mod(p)$.

Donc $m^t - m$ est multiple de p et de q donc de n d'où la conclusion.

Il n'y a que l'utilisateur B qui connaît le nombre d , il est donc le seul à pouvoir calculer l'application D . Pour qu'un autre utilisateur puisse retrouver ce nombre d et ainsi être en mesure de décrypter les messages destinés à B il faudrait qu'il puisse retrouver la décomposition en facteur premier du nombre N (il aurait donc la connaissance de p et q donc de $(p-1) \times (q-1)$ et finalement il pourrait calculer l'inverse de e dans $(\mathbb{Z}/(p-1) \times (q-1)\mathbb{Z})^*$ avec l'algorithme Euclide-Etendu et retrouver d). La sûreté de cette méthode de cryptage vient du fait que la décomposition en facteur premier de l'entier N est impossible à réaliser en un temps raisonnable si p et q sont bien choisis (il faut, entre autres propriétés, que p et q soient de très grands nombres premiers mais ce n'est pas suffisant. Il y a d'autres précautions à prendre pour garantir la solidité de ce chiffrement mais nous ne les aborderons pas dans ce cours).

Un utilisateur mal intentionné pourrait aussi décrypter le message c en tentant de résoudre dans $\mathbb{Z}/N\mathbb{Z}$ l'équation $x^e \equiv c(N)$. Là non plus on ne connaît pas de méthode simple (et tester tous les entiers de 0 à $N-1$ est évidemment beaucoup trop long si N est très grand, de l'ordre de plusieurs milliers de bits).

Exemple:
$p = 7, q = 11$ et $e = 13$

1. Calculez N
2. Calculez d
3. On veut envoyer $m = 9$. Calculez $c = E(m)$.
4. Vérifiez que $D(c) = 9$.

4.3) El Gamal:

Bob veut permettre à ses interlocuteurs (dont Alice) de communiquer avec lui en toute sécurité.

Il dispose d'un nombre premier p (très grand, encodé sur plusieurs milliers de bits) et d'un élément $a \in (\mathbb{Z}/p\mathbb{Z})^*$. On note $q = \omega(a)$ (donc $a^q \equiv 1\,(p)$). Pour l'efficacité du cryptage il faut que q soit un très grand nombre premier tout comme p. En effet le problème du logarithme discret est plus compliqué à résoudre si q est un grand nombre premier.

Les messages "en clair" seront des éléments de $(\mathbb{Z}/p\mathbb{Z})^*$. Bob choisit aléatoirement un nombre entier x tel que $0 < x < q$ et il calcule $h = a^x\,(p)$. Il transmet la clé publique $k_{pub} = (p, q, a, h)$. Si p et a sont bien choisis retrouver x à partir de h est un problème difficile (cf. logarithme discret). L'entier x constitue la clé privée.

Alice veut envoyer le message "en clair" m ($m \in (\mathbb{Z}/p\mathbb{Z})^*$) à Bob. Elle choisit d'abord aléatoirement un entier r tel que $0 < r < q$ puis elle calcule successivement $c_1 = a^r$ et $c_2 = m \times h^r$. C'est le couple (c_1, c_2) qu'elle transmet à Bob.

Pour déchiffrer Bob calcule $\frac{c_2}{c_1^x} \bmod p$ (pour calculer l'inverse de c_1^x dans $(\mathbb{Z}/p\mathbb{Z})^*$ Bob peut calculer c_1^{q-x}).

En effet $\frac{c_2}{c_1^x} \bmod p \equiv c_2 \times c_1^{q-x} \equiv m \times h^r \times (a^r)^{q-x} \equiv m \times a^{rx} \times a^{rq-rx} \equiv m \times (a^q)^r \equiv m \bmod \text{p}$.

Interêt d'El Gamal par rapport à RSA:

Supposons qu'une autre personne veuille transmettre le même message m à Bob. C'est le cas par exemple si Bob récupère les résultats d'un vote: beaucoup de votants auront le même choix à transmettre à Bob, c'est à dire le même message m en clair, mais voudront que ce choix reste secret. Or avec l'algorithme RSA tous ces votants transmettront le même message chiffré c et seront donc facilement identifiables. Avec l'algorithme El Gamal chacun de ces votants transmettra à Bob un message chiffré différent puisqu'il dépend du choix aléatoire et secret de l'entier r . Il sera donc impossible de savoir pour qui chaque personne a voté juste en regardant le message chiffré. Par contre, le chiffrement, le déchiffrement et la transmission sont plus longs avec El Gamal comparés à RSA.

Remarque pratique:

Nous avons vu que trouver $q = \omega(a)$ dans $(\mathbb{Z}/p\mathbb{Z})^*$ n'est pas simple si p est un nombre premier "très grand". On sait que $q = \omega(a)$ divise $\varphi(p) = p - 1$, donc, si $p - 1$ admet peu de diviseurs et des diviseurs simples à trouver, on pourra déterminer facilement $q = \omega(a)$. Dans la pratique on cherche un nombre premier q très grand tel que $p = 2q + 1$ soit aussi premier (on verra comment faire dans le chapitre suivant et en TP). Dans ce cas $\varphi(p) = p - 1 = 2q$ (puisque p est premier) et admet pour diviseurs $1, 2, q$ et $2q$. Pour trouver $a \in \{2, 3, \ldots, q-1\}$ tel que $\omega(a) = q$ on peut exécuter l'algorithme suivant:

1. on choisit au hasard $a \in \{2, 3, \ldots, q-1\}$
2. On calcule $a^2 \bmod \text{p}$:
   (a) si $a^2 \equiv 1\,(p)$ on recommence au point 1
   (b) sinon (donc forcément $a^{2q} \equiv 1\,(p)$) on renvoie $a^2\,(p)$ (c'est à dire le reste de la division euclidienne de a^2 par p).

Annexe: test de primalité:

Propriété 15:

Pour tout $n \in \mathbb{N}^*$ notons $\pi(n)$ le nombre de nombres premiers inférieurs ou égaux à n . Par exemple $\pi(13) = 6$.
Alors $\lim_{n\to+\infty} \frac{\pi(n)}{n/\ln(n)} = 1$.

Donc, pour n assez grand, $\frac{n}{\ln(n)}$ est une estimation de $\pi(n)$, ce qui signifie qu'en choisissant au hasard un entier inférieur ou égal à n on a une probabilité de $1/\ln(n)$ environ d'obtenir un nombre premier. Il faudra en moyenne $\ln(n)$ essais pour tomber par hasard sur un nombre premier (on peut diviser par deux cette probabilité en ne choisissant que des nombres impairs). Pour $n = 2^{1000}$ on peut trouver un nombre premier de $\beta = 1000$ bits en faisant en moyenne $\frac{1}{2}\ln\left(2^{1000}\right) \simeq 347$ essais. Il faut bien sûr un moyen de tester si un nombre entier est premier sans reprendre l'algorithme de décomposition des nombres entiers qui n'est pas polynomial. La définition suivante ainsi que la remarque donne une réponse "presque satisfaisante".

Définition 13:

Un nombre entier $n \in \mathbb{N}^*$ est dit pseudo-premier de base a si n n'est pas premier et $a^{n-1} \equiv 1 \bmod n$.

exercice:
Vérifiez que 341 est pseudo-premier de base 2.

Remarque:

Les nombres entiers n qui ne vérifient pas l'équation $a^{n-1} \equiv 1 \bmod n$ pour un entier $a \in \{2, 3, \ldots, n-1\}$ ne sont donc pas premiers (d'après le petit théorème de Fermat). Les autres sont soit premiers, soit pseudo-premiers de base a.

Pour $a = 2$ la probabilité de tomber sur un nombre pseudo-premier de base 2 est très faible (il n'y a que 22 nombres pseudo-premiers de base 2 inférieurs à 10000). On peut montrer que la probabilité de tomber sur un nombre pseudo-premier de base 2 en prenant au hasard un nombre de β bits tend vers 0 lorsque $\beta \to \infty$. Plus concrètement, un nombre entier de 512 bits choisi au hasard a une probabilité inférieure à $\frac{1}{10^{20}}$ d'être pseudo-premier de base 2 ($\frac{1}{10^{41}}$ pour 1024 bits).

Finalement, pour des grands nombres entiers tirés au hasard, le test qui renvoie "vrai" si $2^{n-1} \equiv 1 \bmod n$ est dans la pratique suffisant pour savoir si ce nombre est premier. On peut encore augmenter la fiabilité de ce test en essayant plusieurs bases a.

(Attention: ce test peut s'avérer insuffisant si les entiers ne sont pas choisis au hasard).