Université Claude Bernard Lyon 1
Polytech'Lyon - Info 5A

Cours de cryptographie
Enseignant : Gérald Gavin
Email : gavin@univ-lyon1.fr

Recueil d'exercices

1 Quelques exercices d'arithmétique modulaire

Exercice 1

Il s'est ecoulé 210476 heures depuis le 1/01/2010. Quelle heure est-il ?

Exercice 2

La relation $\equiv_n$ sur $\mathbb{Z}$ est définie par $a \equiv_n b$ (on pourra noter $a \equiv b \mod n$) si et seulement s'il existe un entier $k \in \mathbb{Z}$ tel que a = b + kn. Montrez que $\equiv_n$ est une relation d'équivalence.

Exercice 3

Soit $a \in \mathbb{Z}$. Montrer que $a\%n \equiv a \mod n$ ($a\%n$ étant le reste de la division de a par n).

Exercice 4

Supposons que $a_1 \equiv b_1 \mod n$ and $a_2 \equiv b_2 \mod n$. Montrer que :

1. $a_1 + b_1 \equiv a_2 + b_2 \mod n$
2. $a_1 b_1 \equiv a_2 b_2 \mod n$

Exercice 5

Soit $a \in \mathbb{Z}$. On note $\overline{a}_n$ la classe à laquelle appartient a. Montrer que $\mathbb{Z}_n = \{\overline{0}_n, \overline{1}_n, \ldots, \overline{n-1}_n\}$.

Exercice 6

On définit les opérations sur $\mathbb{Z}_n$ suivantes :

1. $\overline{a}_n + \overline{b}_n = \overline{a+b}_n$
2. $\overline{a}_n \overline{b}_n = \overline{ab}_n$

Montrez que ces opérations sont bien définies (remarque : on a ainsi défini une structure d'anneau sur $\mathbb{Z}_n$).

Exercice 7

Vérifier que pour tout $x \in \mathbb{Z}_7$, $x^6 \equiv 1 \mod 7$. En déduire que tout élement non-nul de $\mathbb{Z}_7$ est inversible (et donc que $\mathbb{Z}_7$ est un corps).

Exercice 8

Soit $a, n \in Z$. Montrer que si $pgcd(a, n) = 1$ alors $a \not\equiv 1 \mod n$. En déduire que $\overline{a}_n$ n'est pas inversible dans $\mathbb{Z}_n$

Exercice 9

Montrer que $x^5 - x \equiv 0 \mod 30$ n'est pas équivalent à $x^4 - 1 \equiv 0 \mod 30$ ou $x \equiv 0 \mod 30$. On explicitera les 2 ensembles de solution.

Exercice 10

Montrer que $n^5 - n$ est divisible par 30 pour tout $n \in \mathbb{N}$

Exercice 11

Résoudre les équations suivantes :

- $3x \equiv 5 \mod 7$
- $6x \equiv 9 \mod 15$
- $7x \equiv 1 \mod 14$

Exercice 12

Quels sont les éléments inversibles dans $\mathbb{Z}_5$ et dans $\mathbb{Z}_6$?

Exercice 13

Combien y-a-t-il d'entiers n dans $\{1,\ldots,128\}$ tels que $pgcd(n, 100) = 1$?

Exercice 14

Vérifier que la fonction $f : \mathbb{Z}_{15} \to \mathbb{Z}_3 \times \mathbb{Z}_5$ définie par $f(x) = (x \mod 3; x \mod 5)$ est une bijection ? Quelle est la fonction inverse ?

Exercice 15

Combien l'armée de Han Xing comporte-t-elle de soldats si, rangées par 3 colonnes, il reste deux soldats, rangées par 5 colonnes, il reste trois soldats et, rangées par 7 colonnes, il reste deux soldats ?

Exercice 16

Soit $n = pq$ le produit de deux nombres premiers p et q. On définit les sous-ensembles E_p et E_q de $\mathbb{Z}_n$ par $E_p = \{0, p, \ldots, (q-1)p\}$ et $E_q = \{0, q, \ldots, (p-1)q\}$. On note que l'ensemble des éléments non-inversible de $\mathbb{Z}_n$ est égal à $E_q \cup E_q$ et $E_p \cap E_q = \{0\}$. Soit $a, b \in \mathbb{Z}_n$. Quel est le nombre de solutions de l'équation $ax + b = 0$. On distinguera les 16 cas concernant l'appartenance ou non de a et b aux ensembles E_p et E_q. En déduire que si a est non nul, i.e. $a \notin E_p$ ou $a \notin E_q$, alors le nombre de solutions est inférieur à $\max(1/p, 1/q)$.

2 Quelques exercices d'algorithmie...

Exercice 1

Parmi les problèmes suivants, quels sont ceux dont on connait des algorithmes rapides (dont l'exécution prend moins d'un siècle dans le pire des cas avec les meilleurs machines actuelles sur des nombres de taille 1024 bits).

1. Le problème du voyageur de commerce
2. Trouver le plus court chemin dans un graphe.
3. Tester si un entier est premier
4. trouver un nombre premier de taille 1024 bits
5. Factoriser des entiers
6. Calculer le pgcd entre 2 entiers
7. Calculer $a^b \mod n$
8. Le probleme de la sous-somme

Exercice 2

Alice (toujours elle) propose l'algorithme suivant pour choisir un nombre RSA $n = pq$ de taille $k = 2k'$

Entrée : $k = 2k'$

Choisir aléatoirement m dans $\{2^{k'}, ..., 2^{k'+1} - 1\}$

Tant que m n'est pas premier

$m \leftarrow m + 1$

$p \leftarrow m$

$m \leftarrow m + 1$

Tant que m n'est pas premier

$m \leftarrow m + 1$

$q \leftarrow m$

Renvoyer $n = pq$

---

1. Cet Algorithme est-il correct ? En particulier renvoit-il toujours un nombre de taille k ?
2. Soit $k = 1024$. Expliquez pourquoi la probabilité que le nombre renvoyé ne soit pas de taille 1024 est très petite.
3. Expliquez pourquoi cet algorithme ne peut pas être utilisé pour générer des clés RSA (on pourra proposer une méthode efficace qui permette de factoriser un nombre n renvoyé par l'algorithme).

Exercice 3

Soit a, b, c trois entiers et ExpMod l'algorithme suivant :

ExpMod(a, b, c)
$r = 1$
pour $i = 1$ *à* b
  $r = r \times a \mod c$
retourner r

1. Que calcule ExpMod ?
2. Quelle est la complexité de cet algorithme ?
3. Evaluer le temps d'execution si a, b, c sont des entiers de 1024 bits.
4. Proposer un algorithme rapide calculant la même fonction que ExpMod.

Exercice 4

Montrer que s'il existe un algorithme rapide pour calculer $\phi(n)$ alors il existe un algorithme rapide pour factoriser n. Quelles conséquences pour la sécurité de RSA ?

Exercice 5

Soit $n = pq$ le produit de 2 grands entiers. Supposons que l'on dispose d'un algorithme ExtractRacine qui prend en entree $y \in \mathbb{Z}_n$ et qui renvoie $x \in \mathbb{Z}_n$ tel que $x^2 \equiv y \mod n$ si un tel x existe.

1. Soit $y \in \mathbb{Z}_n$. Quel est le nombre possible de racines carrées de y, i.e. une racine carrée étant un nombre $x \in \mathbb{Z}_n$ tel que $x^2 = y$.
2. Montrer (ou constater sur des exemples) que pour tout $y \in \mathbb{Z}_n$, la difference de 2 de ses racines carrées (si y en possède) est un multiple de p.
3. En déduire un algorithme de factorization de n (qui est efficace si ExtractRacine est efficace).

3 Quelques exercices basiques sur RSA/Paillier...

Exercice 1

Soit $pk = (n = pq, e)$ une clé publique RSA. Peut-on encrypter p avec pk ?

Exercice 2

Prouver que RSA est correct ? Autrement dit prouver que pour tout $x \in \mathbb{Z}_n^*$

$$(x^e)^d \equiv x \mod n$$

Exercice 3

Soit $n \in \mathbb{N}$. On note m l'inverse de n dans $\mathbb{Z}/\phi(n)\mathbb{Z}$ (on supposera n inversible). Soit $x, r \in \mathbb{Z}/n\mathbb{Z}$ et $X = (1 + x \cdot n)r^n \mod n^2$.

1. Montrer que :
   — $X \equiv r^n \mod n$
   — $r \equiv X^m \mod n$
   — $x = (X \cdot r^{-n} \mod n^2 - 1)/n$
2. En déduire un cryptosystème.
3. Montrer que ce cryptosystème est homomorphique additif, i.e. $D_{sk}(E_{pk}(x).E_{pk}(y)) = x + y$
4. Proposez une application à cette propriété d'homomorphie.

Exercice 4

Alice a généré une clé publique et secrète du cryptosystème RSA. Bob souhaite lui envoyer un message secret. Pour ceci, il decide de lui envoyer une encryption du code ascii de chaque lettre. Critiquer cette manière de procéder.

Exercice 5

Quel est l'étape la plus longue dans la génération de clés RSA ?

Exercice 6

Nous souhaitons élaborer un mécanisme de signature électronique permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier. Enoncer des propriétés que devrait vérifier ce mécanisme et proposer une solution.

Exercice 7

Alice a généré une clé publique et secrète du cryptosystème RSA. Elle a, en outre, publié une liste de question binaires (dont la réponse est oui/non). Bob souhaite y répondre de manière sécurisé. Proposer un protocole pour réaliser ceci.

Exercice 8

Générer une clé publique et secrète du cryptosystème Paillier et les tester. On vérifiera en outre la propriété d'homomorphie. Dire comment ce cryptosystème pourrait être utilisé pour le vote électronique.

4 Quelques attaques...

Exercice 1

Pour des besoins cryptographiques non explicités ici, Alice doit générer un grand nombre T de clés RSA différentes, i.e. $(pk_1 = \{n_1, e_1\}, sk_1 = \{d_1\})$,...,$(pk_T = \{n_T, e_T\}, sk_T = \{d_T\})$. Afin de gagner du temps, elle décide de ne générer que $t = O(\sqrt{T})$ nombres premiers $p_1, ..., p_t$. On considère l'ensemble N défini par $N = \{p_i p_j \mid (i,j) = \{1, ..., t\}^2, i \neq j\}$

1. Montrer que si $t = 2\sqrt{T}$ alors le cardinal de N est supérieur à T.
2. Posons $t = 2\sqrt{T}$. Alice choisit arbitrairement T éléments différents $n_1, ..., n_T$ dans N et génère les clés $(pk_1 = \{n_1, e_1\}, sk_1 = \{d_1\})$,...,$(pk_T = \{n_T, e_T\}, sk_T = \{d_T\})$
   (a) Comment Alice choisit-elle les valeurs $(e_i, d_i)_{i=1,...,T}$ pour que les clés soient correctes.
   (b) Dire pourquoi cette facon de générer les clés s'avère avantageuse en termes de temps de calcul. En vous servant de l'expérience acquise en TP et éventuellement d'arguments plus formels, évaluer le gain de cette méthode par rapport la méthode (classique) consistant à exécuter T fois la fonction KeyGen.
   (c) Montrer que cette méthode n'est pas judicieuse en terme de sécurité (pensez à Euclide...).

Exercice 2

Une voyante australienne réputée prétend avoir deviné les 6 numeros gagnants (compris entre 1 et 49) du prochain tirage du loto. Pour des raisons que nous n'expliciterons pas ici, elle souhaite vous en faire profiter (et seulement vous). Etant d'un naturel peu partageur, vous souhaitez sécuriser la communication. Pour ceci, vous décidez d'utiliser le cryptosysteme RSA en transmettant à cette voyante une clé publique (n, e) de taille 1024.

1. Sans consigne de votre part, mais quand même familiarisée avec RSA et la classe BigInteger, la voyante vous envoie une encryption de chacun des 6 numeros. Expliquez pourquoi cette façon de procéder n'est absolument pas sécurisée.
2. Afin de pallier la faille précedente, vous suggérez à la voyante de vous envoyer une encryption de la concaténation de ces 6 numeros (donnant un nombre à au plus 12 chiffres). En considérant qu'un attaquant dispose de ressources lui permettant d'effectuer une exponentiation modulaire $x^e \mod n$ en 100ms pour tout $x \in \mathbb{Z}_n$, estimez le temps nécéssaire (en moyenne) à l'attaquant pour retrouver les 6 numéros.
3. Dans l'euphorie, vous avez malencontreusement choisi $e = 3$. Pourquoi un tel choix n'est-il pas judicieux ? Proposer une attaque quasi-instantanée permettant à l'attaquant de retrouver les 6 numéros (on utilisera le fait que le message encrypté est trop petit par rapport à n).

Exercice 3

On suppose que Bob chiffre successivement avec RSA pour Alice deux messages de la forme m et $m+\delta$. Les messages chiffrés correspondants sont c_1 et c_2 (comme dans le fichier qui contient les challenges).

1. Combien vaut :
$$\frac{(m+1)^3 + 2m^3 - 1}{(m+1)^3 - m^3 + 2} \mod N?$$
2. Sachant que l'exposant public d'Alice est $e = 3$ et que $\delta = 1$, utiliser la relation précédente pour retrouver le message m
3. Implémenter cette attaque.

Exercice 4

L'échange de clés de Diffie-Hellman fonctionne de la façon suivante : Alice et Bob se mettent d'accord sur un groupe G d'ordre premier p, et choisissent un générateur g (tous les éléments de ce groupe sont donc de la forme g^x, avec x un entier compris entre 1 et $p-1$). De son côté, Alice choisit un entier a compris entre 1 et $p-1$, et calcule g^a. Bob, quant à lui, choisit un entier b compris entre 1 et $p-1$, et calcule g^b. Alice et Bob s'échangent les éléments g^a et g^b, et conservent secrets a et b. Ils peuvent ainsi calculer tous les deux la clé $K = g^{ab}$.

1. A quoi peut servir un tel protocole d'échange de clés ?
2. Rappelez comment Alice et Bob calculent chacun la clé K.
3. A quelle condition algorithmique ce protocole est-il sûr ?
4. Plus pratiquement, Alice et Bob utilisent des clés de la forme $g^x(\mod p)$ avec $g = 2$ et $p = 29$.
5. La clé publique d'Alice est 15. Vous jouez le rôle de Bob : choisissez une clé secrète, calculez la clé publique correspondante, et donnez la clé secrète partagée entre Alice et vous.
6. Proposez une généralisation de l'échange de clés Diffe-Hellman à 3 personnes.
7. Considérons maintenant l'échange de clés suivant :
   (a) Alice tire deux suites binaires aléatoires de longueur n : k et $r \in \{0,1\}^n$. Elle envoie à Bob $s = k \oplus r$.
   (b) Bob tire aléatoirement une suite binaire $t \in \{0,1\}^n$ et envoie $u = s \oplus t$ à Alice.
   (c) Alice calcule $w = u \oplus r$ et renvoie w à Bob.
   (d) Alice choisit la clé k comme clé partagée.

   Montrez que Bob peut également calculer la clé k à partir des informations reçues. Montrez cependant que ce protocole d'échange de clés n'est pas sûr.

Exercice 5

1. Rappelez le fonctionnement du chiffrement RSA.
2. Expliquer pourquoi il est dangereux d'attribuer deux couples clé publique/clé secrète (e_1, d_1) et (e_2, d_2) associé chacun à un même module n à deux utilisateurs différents (un chiffré pour l'un peut-il être lu par l'autre ?).
3. Supposons que ces deux utilisateurs se fassent mutuellement confiance et qu'un même message m soit envoyé à ces deux utilisateurs : vous allez montrer qu'un espion qui voit passer les deux chiffrés pour ces deux utilisateurs peut retrouver le message m sans les clés secrètes. Pour vous aider, considérons l'exemple suivant. Soient $n = 91$, $(e_1, d_1) = (11, 59)$ et $(e_2, d_2) = (5, 29)$ les paires de clés publique/privée de ces deux utilisateurs. Soient $u = 1$ et $v = -2$: combien vaut $e_1 \times u + e_2 \times v$? Calculez les chiffrés c_1 et c_2 du message $m = 2$, et calculez $c_1^u \times c_3^v (\mod n)$. En vous inspirant de ces calculs, donnez une description formelle de l'attaque de RSA dans ce cas pathologique, et montrez qu'elle peut se réaliser en temps polynomial avec les seules données publiques (le pgcd de e_1 et e_2) est à prendre en compte).

5 Quelques protocoles...

Exercice 1

Imaginons qu'Alice (resp. Bob) possède un vecteur secret (à coordonnées entières supposées plus petites qu'une certaine valeur A) $\vec{u}$ (resp. $\vec{v}$). Etablir un protocole ProdScal permettant de calculer le produit scalaire $\vec{u}\vec{v}$ sans révéler aucune information sur $\vec{u}$ et $\vec{v}$ (autre que $\vec{u}\vec{v}$). Analyser la sécurité de ce protocole.

Exercice 2

Alice possède une 3-DNF secrète f construite à partir de T variables booléennes. Bob possède T variables booléennes secrètes $x_1, ..., x_T$. Construire un protocole EvalDNF permettant à Bob d'obtenir $f(x_1, ..., x_T)$ sans rien révéler sur $x_1, ..., x_T$ et sans rien apprendre sur f. Analyser la sécurité de ce protocole.

Exercice 3

Alice propose une liste de 10 questions a Bob et n'accepte de répondre qu'a une seule d'entre elle. Bob, quant à lui, souhaite que sa question reste secrète. Autrement dit, Bob choisit secrètement une des 10 questions et souhaite obtenir une réponse sans qu'Alice ne sache quelle question a été choisie et Alice veut avoir la garantie que Bob n'a obtenue qu'une seule réponse. Etablir, si possible, un protocole réalisant ceci.

Exercice 4

Proposer une solution pour la distribution de carte virtuelle + analyser + tout autre amélioration ou extension. Concrètement, 4 joueurs en ligne souhaitent se distribuer 8 cartes (pour jouer à la belote par exemple). Chaque joueur devra être assurer qu'aucun autre joueur (ou groupe de joueurs) ne peut tricher, i.e. influencer la distribution, connaitre ses cartes, etc...

Exercice 5

Considérons le cryptosystème de Paillier, à savoir les fonctions Paillier.KeyGen, Paillier.Encrypt et Pailler.Decrypt. Pour simplifier les notations, $[x]_{pk}$ (ou simplement $[x]$ lorsqu'il n'y aura aucune ambiguité sur la clé publique) désignera une encryption d'une valeur x avec la clé publique pk.

Nous supposerons qu'Alice a généré un couple de clés $(pk, sk) = (n, \phi(n))$ avec la fonction Paillier.KeyGen. Nous supposerons en outre que Bob dispose de deux encryptions X et Y de deux valeurs[1] x et y i.e. $X = [x]$ et $Y = [y]$. On supposera que x et y sont inconnues de Bob et d'Alice (on pourra supposer qu'elles proviennent d'un protocole antérieur ou d'une tierce partie). L'objet de ce CC est d'établir un protocole entre Bob et Alice permettant à Bob d'obtenir une encryption du produit[2] $xy \mod n$ tout en garantissant que les valeurs x, y ne soient dévoilées ni à Bob ni à Alice. Un tel protocole, appelé Multiplication, vous est fourni ci-dessous dans une version semi-detaillée.

Multiplication

1. Autrement dit, Paillier.Decrypt$(sk, X) = x$ et Paillier.Decrypt$(sk, Y) = y$.
2. Les propriètés homomorphiques seules de Paillier ne permettent pas à Bob de le faire en local.

Pré-requis : *Bob possède deux encryptions $[x]$ et $[y]$ de deux valeurs x et y supposées n'être connues ni de Bob ni d'Alice.*

1. **Bob** envoie des encryptions de $r + x$ et de $s + y$ où r et s sont choisis aléatoirement (par Bob) dans $\mathbb{Z}/n\mathbb{Z}$
2. **Alice** génère et envoie une encryption de $(x + r)(y + s)$.
3. **Bob** génére et retourne une encryption de $(x + r)(y + s) - sx - ry - rs$.

Rappel des propriètés homomorphiques du cryptosystème de Paillier. Soient $u, v \in \mathbb{Z}/n\mathbb{Z}$, $\mathsf{Paillier.Decrypt}([u][v] \mod n^2) = u + v \mod n$ et $\mathsf{Paillier.Decrypt}([u]^v \mod n^2) = uv \mod n$.

Q1 A quels ensembles appartiennent les messages et les encryptions considérés par le cryptosystème de Paillier ?

Q2 Dire par qui et comment sont utilisées les propriétés homomorphiques additives de Paillier dans ce protocole.

Q3 Ecrire le protocole Multiplication de manière la plus détaillée possible (en utilisant les fonctions Paillier.KeyGen, Paillier.Encrypt et Paillier.Decrypt).

Q4 Pourquoi Alice n'apprend-elle rien sur x, y si Bob respecte le protocole ?

Q5 Pourquoi Bob n'apprend-il rien sur x, y si Alice respecte le protocole ?

Q6 En supposant que Bob et Alice respectent le protocole, justifier le fait que l'encryption retournée par Bob encrypte $xy \mod n$.

Q7 Expliquer comment Alice peut agir (en ne respectant pas le protocole) pour que l'encryption retournée par Bob n'encrypte pas le produit $xy \mod n$.

Q8 Pourquoi ce protocole ne peut-il pas s'implémenter avec El Gamal (dans sa version homomorphique additif) ?

6 Se tester...

On génère une paire de clés RSA $pk_{RSA} = (N, e)$ et $sk_{RSA} = (d, p, q)$ et une paire de clés ElGamal $pk_{EG} = (p', g, h)$ et $sk_{EG} = x$ chacune de **4096** bits. Parmi les affirmations suivantes, dire (en justifiant précisément votre réponse) celles qui sont vraies et celles qui sont fausses.

1. e peut être choisi égal à 6.
2. Tous les éléments non-nuls de $\mathbb{Z}/N\mathbb{Z}$ sont inversibles.
3. La proportion d'éléments non-inversibles de $\mathbb{Z}/N\mathbb{Z}$ est inférieure à 1%.
4. p' est premier.
5. Il existe un algorithme rapide (de complexité polynomiale) qui prend en entrée N et qui retourne un élément $x \in (\mathbb{Z}/N\mathbb{Z}) \setminus \{0\}$ non inversible.

6. Soit a, b, c trois entiers et considérons l'algorithme A suivant :

   $\mathsf{A}(a,b,c)$
   $R = 1$
   Pour i $= 1$ à b faire
   $\quad R = R \times a \pmod{c}$
   Retourner R

   L'algorithme A peut être utilisé dans la fonction de déchiffrement $\mathsf{Decrypt}_{RSA}$.

7. Quand on chiffre avec Elgamal deux fois le même message m pour la même personne possédant la clé publique pk_{EG}, les deux chiffrés sont égaux.
8. Si $c = \mathsf{Encrypt}_{RSA}(pk_{RSA}, m)$ et $c' = \mathsf{Encrypt}_{RSA}(pk_{RSA}, m')$. L'élément $c \times c' \pmod{N}$ est un chiffré de $m \times m' \pmod{N}$.
9. Soit r un entier dans $\{1, \ldots, p' - 1\}$. On connait un algorithme rapide qui prend en entrée $g^r \pmod{p'}$ et qui renvoie r.
10. Si la seule méthode de factorisation existante était la méthode des divisions successives, la taille d'un module RSA pour atteindre un niveau de sécurité de 80 bits serait de 160 bits.
11. $33^{33} \pmod{7} \equiv 5^3 \pmod{7}$.
12. 12 n'est pas inversible dans $\mathbb{Z}/39\mathbb{Z}$.
13. Soient a, b deux entiers positifs de k bits. Il existe un algorithme rapide (polynomial en k) pow tel que $\mathsf{pow}(a, b) = a^b$.
14. Soient a, b, c trois entiers positifs de k bits. Il existe un algorithme rapide modPow tel que $\mathsf{modPow}(a, b, c) = a^b \mod c$.
15. Soit $x \in (\mathbb{Z}/N\mathbb{Z})^*$. L'inverse modulaire de x est égal à $y = x^{\varphi(N)-1} \mod n$.
16. Si e est un entier de 32 bits alors d est un entier de plus 32 bits.

TP 1

L'objectif de se TP est d'expérimenter quelques notions d'arithmétique modulaire et de découvrir la classe BigInteger *de Java. Tous les entiers manipulés dans ce TP seront des objets de cette classe.*

1. Compter le nombre d'éléments inversibles dans $\mathbb{Z}_{377}$. Le resultat vous surprend-il ?
2. Implémenter un test de primalité naif.
3. Implémenter le test de fermat
4. Combien d'erreurs commet le test de Fermat sur les 10000 premiers entiers ¿2
5. A partir de quelle taille d'entree, votre test naif devient plus lent que le test de Fermat ?
6. Générer 2 entiers premiers p et q de 512 bits.
7. Calculer n=pq et $\phi_n = (p-1)(q-1)$.
8. Générer aléatoirement un nombre e de 16 bits qui soit premier avec ϕ_n.
9. Calculer l'inverse modulaire d de e modulo ϕ_n. Quelle relation vérifient e et d ? La vérifier.
10. Pour plusieurs éléments $x \in \mathbb{Z}_n$, calculer $X = x^e \mod n$ et $X^d \mod n$. Que constate-t-on ?
11. Déduire de ce qui précède un cryptosystème à clé publique. On explicitera précisément les fonctions KeyGen, Encrypt et Decrypt. Quelle est la complexité de ces fonctions ?
12. Dire sur quoi pourrait reposer la sécurité de ce cryptosystème ?
13. On imagine le protocole suivant. Alice génère un couple de clés privée/publique et publie la clé publique avec el cryptosystème précédent. Alice pose des questions binaires (réponse par oui ou par non) à Bob qui lui répond en envoyant une encryption de 1 si la réponse est oui et une encryption de 2 sinon. Dire en quoi ce protocole n'est pas sure. Proposer amélioration (en terme de sécurité) de ce protocole.

TP 2

Vous avez implémenté le cryptosystème de Paillier, à savoir les fonctions Paillier.KeyGen, Paillier.Encrypt et Pailler.Decrypt. On aura aussi besoin de la fonction Paillier.DecryptPlus (légère modification de Paillier.Decrypt) définie par :

Paillier.DecryptPlus prend en entrée une encryption $X = (1 + xn)r^n \mod n^2$ et retourne (x, r).

Pour simplifier les notations, $[x]_{pk}$ (ou simplement $[x]$ lorsqu'il n'y aura aucune ambiguité sur la clé publique) désignera une encryption d'une valeur x avec la clé publique pk.

Nous supposerons dans tout le TP qu'Alice a généré un couple de clés $(pk, sk) = (n = pq, \phi(n))$ avec la fonction Paillier.KeyGen. Nous supposerons en outre que Bob dispose de deux encryptions X et Y de deux valeurs [3] x et y i.e. $X = [x]$ et $Y = [y]$. On supposera que x et y sont inconnues de Bob et d'Alice (on pourra supposer qu'elles proviennent d'un protocole antérieur ou d'une tierce partie). L'objet de ce TP est d'établir un protocole entre Bob et Alice permettant à Bob d'obtenir une encryption du produit [4] $xy \mod n$ tout en garantissant que les valeurs x, y ne soient dévoilées ni à Bob ni à Alice. Un tel protocole, appelé Multiplication, vous est fourni ci-dessous dans une version semi-detaillée.

Multiplication

Pré-requis : *Bob possède deux encryptions X et Y de deux valeurs inconnues x et y.*

1. **Bob** envoie des encryptions de $r + x$ et de $s + y$ où r et s sont choisis aléatoirement (par Bob) dans $\mathbb{Z}/n\mathbb{Z}$
2. **Alice** génère et envoie une encryption de $(x + r)(y + s)$.
3. **Bob** génére et retourne une encryption de $(x + r)(y + s) - sx - ry - rs$.

Q1 Implémenter le protocole Multiplication. Pour cela, on créera 3 méthodes mult1, mult2 et mult3 correspondant respectivement aux étapes 1,2 et 3 du protocole. multk prendra en entrée que ce qui est connu par la partie exécutante (Alice ou Bob) au début de l'étape k. Pour simplifier l'implémentation, "*envoyer à*" sera juste remplacé par "*retourner*". Par exemple mult1 pourra prendre en entrée pk, X, Y et retournera une encryption de $x + r$ et une de $y + s$.

Si Alice dévie du protocole à l'étape 2 en n'envoyant pas une encryption de $(x + r)(y + s)$ alors Bob retournera une encryption incorrecte à l'étape 3. Nous proposons de construire un protocole MultiProof qui va permettre à Bob de vérifier qu'Alice a bien respecté l'étape 2.

Plus généralement, supposons qu'Alice (qui possède la clé secrète) dispose de 3 encryptions $[\alpha]$, $[\beta]$ et $[\gamma]$ tel que $\gamma = \alpha\beta \mod n$. Elle souhaite prouver à Bob (appelé le vérifieur) que $\gamma = \alpha\beta \mod n$ sans révéler quoique que ce soit sur ces valeurs à Bob. A l'issue du protocole, Bob devra être certain que cette relation est vérifiée (si elle ne l'est pas, le protocole devra échouer). Nous vous proposons le protocole suivant :

---

3. Autrement dit, Paillier.Decrypt$(sk, X) = x$ et Paillier.Decrypt$(sk, Y) = y$.
4. Les propriètés homomorphiques seules de Paillier ne permettent pas à Bob de le faire en local.

MultiProof

Pré-requis : *Alice propose 3 encryptions* $[\alpha]$, $[\beta]$ *et* $[\gamma]$ *tel que* $\gamma = \alpha\beta \mod n$. *Ces 3 encryptions sont évidemment connues de Bob. Alice souhaite prouver que* $\gamma = \alpha\beta \mod n$ *sans rien devoiler sur ces valeurs. La preuve sera acceptée si le protocole suivant n'échoue pas.*

1. **Alice** choisit δ aléaoirement dans $\mathbb{Z}/n\mathbb{Z}$ et envoie les encryptions $[\delta]$ et $[\pi]$ à Bob où $\pi = \delta\beta \mod n$.
2. **Bob** choisit e aléatoirement dans $\mathbb{Z}/n\mathbb{Z}$ et l'envoie à Alice.
3. **Alice** calcule :
   — $(a, r) \leftarrow \mathsf{Paillier.Decryptplus}([\alpha]^e[\delta] \mod n^2)$
   — $(a', r') \leftarrow \mathsf{Paillier.Decryptplus}([\beta]^a[\pi]^{-1}[\gamma]^{-e} \mod n^2)$
   et envoie (a, r) et (a', r') à Bob.
4. **Bob** vérifie que :
   (a) $(1 + an)r^n \equiv [\alpha]^e[\delta] \mod n^2$
   (b) $(1 + a'n)r'^n \equiv [\beta]^a[\pi]^{-1}[\gamma]^{-e} \mod n^2$
   (c) $a' = 0$
   Si au moins l'une des trois vérifications échouent alors le protocole échoue.

Q2 Montrer que $a = \alpha e + \delta$ et $a' = 0$ si Alice respecte le protocole.

Q3 Déduire de la question précédente que si Alice respecte le protocole alors Bob n'apprend rien sur α, β, γ (on admettra que r, r' sont independants de α, β, γ et que Paillier (son cryptosystème) est sémantiquement sûr). On utilisera le fait que δ est choisi indépendamment de α.

Q4 Pourquoi les vérifications 4.a et 4.b permettent à Bob de vérifier que $[\alpha]^e[\delta] \mod n^2$ encrypte a et que $[\beta]^a[\pi]^{-1}[\gamma]^{-e} \mod n^2$ encrypte a' ?

Q5 Montrer que si $\gamma \neq \alpha\beta \mod n$ et si les vérifications 4.a et 4.b n'échouent pas alors $a' = 0$ avec une probabilité négligeable (même si $\pi \neq \delta\beta \mod n$). Vous pouvez essayer de montrer qu'elle est inférieure à $\max(1/p, 1/q)$ en utilisant le fait que e est choisi aléatoirement dans $\mathbb{Z}/n\mathbb{Z}$ et que $a' = (\alpha e + \delta)\beta - \pi - e\gamma \mod n$ (voir exercice 1.16).

Q6 Déduire de la question précédente que si $\gamma \neq \alpha\beta \mod n$ alors le protocole échoue avec une probabilité proche de 1.

Q7 Comment MultiProof peut-il être utilisé dans Multiplication pour sécuriser l'étape 2 ?

Q8 Implémenter MultiProof. On créera 4 méthodes MultiP1, MultiP2, MultiP3, MultiP4 en adoptant les mêmes conventions que celles utilisées dans Q1.