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).

8.5 Quelques protocoles...

Exercice 104. *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 105. *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 106. *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 107. *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 108. *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*

1. Autrement dit, Paillier.Decrypt$(sk, X) = x$ et Paillier.Decrypt$(sk, Y) = y$.

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

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.

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 :

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 :
   — $(a, r) \leftarrow$ Paillier.Decryptplus$([\alpha]^e[\delta] \mod n^2)$
   — $(a', r') \leftarrow$ 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.

Projet

Vous trouverez ici quelques idées/propositions de projet. Il s'agira de proposer, d'analyser et d'implémenter un ou plusieurs protocoles de calcul multi-parties. Toute originalité est la bienvenue. Ce projet donnera lieu à un mini-rapport qui servira de base à un oral d'évaluation.

Q1 Le protocole Multiplication vu au chapitre 7 n'est pas sécurisé, car à l'étape 2, Bob peut dévier du protocole en envoyant une encryption W de son choix. Le TP 2 permet de résoudre ce problème. Il propose, en effet, une preuve intéractive sans divulgation de connaissance (zero-knowledge proof) permettant à Bob de prouver qu'il ne dévie pas du protocole. Il vous est demandé de réaliser ce TP et de proposer une implémentation sécurisée de Multiplication.

Q2 Exercice 105.

Q3 Après de nombreuses années de mariage cryptographique, Bob a eu un n^{ieme} comportement malveillant vis à vis d'Alice lors d'un protocole houleux. Une mesure d'éloignement à été prononcée contre lui. Pour garantir ceci, il est affublé d'un bracelet numérique connecté (muni d'une petite capacité de calcul) et géolocalisé. Le "bracelet" connait sa position, i.e. ses coordonnées (x_B, y_B) dans un repère orthonormé arbitraire dont l'unité est le mètre. On supposera en outre que x_B et y_B sont des entiers. Il s'agira d'établir un protocole entre Alice (qui connait sa propre position $(x_A, y_A) \in \mathbb{N}^2$) et le bracelet, permettant à Alice de savoir, quand elle le souhaite, si Bob est à moins de 100 mètres ou non (et rien d'autre sur la position de Bob). On pourra aussi réflechir à la variante suivante. Si Bob est à moins de 100 mètres alors (x_B, y_B) est révélé à Alice (et n'est pas révélé sinon). On supposera que Bob (via son bracelet electronique) est honnête et qu'il respecte le protocole, e.g. le protocole est implémenté en hard sur son bracelet.