Devoir à envoyer au plus tard le 18 mai à quentin.deschamps@univ-lyon1.fr. Ne pas oublier de joindre les algorithmes.

1 Fonction de hachage

1. Rappelez les trois propriétés que doit vérifier une fonction de hachage.
2. Soit f la fonction $x \to x^2 \mod N$. Quelle propriétés vérifie f ?
   Soit $g : \{0,1\}^* \to \{0,1\}^n$ que l'on suppose résistante à la collision. Soit $h : \{0,1\}^* \to \{0,1\}^{n+1}$ définie par
   - $h(x) = 1|x$ si x est de longueur n.
   - $h(x) = 0|g(x)$ sinon

   Quelle propriétés vérifie h ?
3. Quels autres implications ou non-implications pouvez-vous trouver.
4. Soit $h : \{0,1\}^* \to \{0,1\}^n$. On suppose que pour un élément m aléatoire, la probabilité d'avoir une valeur donnée $h(m)$ est de $\frac{1}{2^n}$ (l'espace d'arrivée est équiprobable). Donner une attaque pour chaque propriété de h ainsi que sa complexité.

2 A propos de RSA

1. Bob et Charles ont pour clé publique RSA respectivement (N, e_1) et (N, e_2) avec e_1 et e_2 premiers entre eux. Alice envoie le même message m crypté par les clés publiques RSA de Bob et Charles en c_1 et c_2. Expliquer comment Eve, qui intercepte les deux messages cryptés et qui connaît les clés publiques de Bob et Charles, peut retrouver le message clair m.
2. Le système AES nécessite des clefs secrète de 256bits. Combien de bits doivent être communiqués avec RSA pour obtenir une clef de 256 bits de façon sécurisée ? Même question en utilisant le protocole Diffie-Hellman.
3. De nombreuses attaques ont été réalisées contre les cartes bancaires dans les années 90. Recherchez quel principe de cryptographie n'avait pas été respecté dans le protocole.

3 Algorithme de factorisation

3.1 Consignes générales

Le but de cette partie est de concevoir un algorithme de factorisation dans le but de casser des clefs RSA. Il faudra donc implémenter différentes méthodes de résolution ainsi qu'une méthode de sélection pour savoir quel algorithme utiliser en fonction de l'entier à factoriser. Le code est demandé en Python. L'utilisation de librairie Python ou Sage est autorisée (voir recommandée dans le cas de l'algorithme de Dixon) dans les limites du raisonnable. Pour tester vos codes : https://sagecell.sagemath.org/. Permet de tester des codes Python et Sage si le temps d'exécution est assez court.
Les algorithmes ont été plus ou moins détaillés en cours, des recherches d'optimisation peuvent être nécessaires pour factoriser tous les nombres mais il n'est pas nécessaire de résoudre tous les cas proposés pour avoir une bonne note, elle ne sera pas proportionnelle au nombre de cas résolus. En revanche la clarté du code est une donnée importante.
Le travail demandé est personnel, comme d'ailleurs pour le reste du DM. Il n'est pas interdit de communiquer entre vous ou d'utiliser internet mais le travail rendu doit être le votre. (En particulier si votre algorithme ne parvient pas à factoriser un nombre test, inutile de demander à quelqu'un d'autre la factorisation)
Il y a quelques questions en plus du code ne les oubliez pas.

3.2 Pour s'échauffer

1. Programmer un algorithme par division successive. Quels nombres arrivez-vous à factoriser en temps raisonnable (quelques secondes).
   Rappel : on cherche à factoriser des modules RSA, de la forme d'un produit de deux nombres premiers de même taille.

2. Avant de factoriser un nombre il vaut mieux vérifier qu'il n'est pas premier. Comment fonctionne la fonction isprime de Python ? Quel est le meilleur algorithme connu en terme de complexité pour tester la factorisation ? Commenter.

3. Programmer un algorithme qui vérifie si un nombre est premier (en utilisant la fonction isprime) ou une puissance d'un nombre premier.

3.3 "Vrais" algorithmes de factorisation

1. Programmer l'algorithme de Pollard-rho

2. (+ Difficile) Programmer la méthode de Dixon

3. Comparer les temps d'exécutions des deux algorithmes en fonction de la taille de l'entrée N en entrée.

3.4 Assemblage des briques

1. En utilisant les algorithmes précédents construire un algorithme de factorisation. L'algorithme doit d'abord tester si l'entier n'est pas facilement factorisable et si il ne l'est pas appliquer la méthode la plus adaptée.

2. Justifier les choix fait pour la sélection des algorithmes.

3.5 Jeu de test

Tous les messages ont été codés de la même façon : convertir les caractères en leur valeur ASCII (en décimal), puis coder des blocs de longueur K avec RSA. Connaissant (N, K, e), déchiffer les messages suivants (écrire un algorithme peut éviter certaines opérations fastidieuses et répétitives)

1. N=11095304447, K=9, e=65537
   M= [7119657469, 8244600096, 7228338652, 439538184, 10872881821, 2615561934, 10510910117, 1034775453, 2661446360, 4380943322, 10510910117, 2458204423, 8630893105, 10673509276, 7171030420, 7699182760, 8491325505, 10670139061, 3910147605, 9047435195, 4563098303, 2264081866, 397778223, 2767783062, 546377399, 263775487, 10192961938]

2. N=297087870763549, K=14, e=65537
   M= [216462957919265, 11360499685802, 58283718045013, 20578508675359, 174502249319757, 231221294836467, 214389560898139, 271160178751541, 279652685760258, 228517283031473, 37142928787286, 107780532741266, 136352004692999, 29964013150189, 265797771818931, 187190758570528, 9932958598209, 29111711590524, 194352973252958, 230830495189635, 212620117025332, 229702290210530, 160973468863924, 23127589801136, 201665521054922, 61600254321379, 248532885876681, 34189803954640]

3. N= 255592102731423299, K=17, e=65537
   M=[207817456905208502, 148237420130970535, 25087225291527285, 233436535648110063, 159392073950854050, 52863254448506362, 138700252138657950, 26340473738589817, 192647457908274285, 159248177184003298, 241878950558575276, 131057396013197036, 254583836697279019, 111051182863360283, 21656896787113568, 242581769384096255, 238709991035940591, 172719480358706047, 77288243533068995, 1182050648744971, 48139955056215262, 157221288411494991, 146310738979984979, 149385443623579598]

4. N=221213045967263709361, K=20,e=65537
   M= [41431523298299409197, 197089725873645785813, 149852739520996827 27, 39734566053797602 08, 167181558987837424665, 23656448330532954908, 30351033817393374306, 61459830460950544807, 120062293853293710561, 133203037195735728438, 8484338822150773069, 219185394266228121321, 122349250409608016299, 80595398234533564404, 104936681771210536097,

14975660402989400329 8, 11760764403927176855 2, 90716286912605415937, 21904136410622221633 4, 44690409448589276378, 40533143215516486 92, 18941964909871427830 7, 18496441931780400572 6, 11405076172724884340 0, 5951741865862548541, 70447219437403430458]

5. N=189543256862584851367919 9, e=3, K=23
M= [128438725742435136729932 2, 85658141972472949349325 0, 18749042910046832227081 38, 65354190768130416522425 7, 29085399897185429591231 4, 18224873195323133651933 6, 65681022181187575107945 5, 18562585167019954820228 1, 10804484955345933787107 04, 74595342412235794859616 1, 16653262267818439040191 92, 63702936075582455374487 2, 16471852289316076980567 29, 17144036847485828624202 68, 14886160858874651369826 72, 85833250326553767453989, 29382392916428174573644 2, 20979396729620356118080, 47219847622749873789747 3, 13182469177901179900518 53]