2.6 Corrections

Solution de l'exercice 13

Il s'agit d'un des principes énoncés par Kerckhoffs dans « La cryptographie militaire ». Ce principe stipule que la sécurité d'un algorithme cryptographique ne doit pas reposer sur le fait que l'algo soit secret mais sur le fait qu'il soit paramétré par une *clef*, ainsi, l'algorithme peut être révélé sans compromettre sa sécurité. En effet, dans le cas où le secret réside (uniquement) dans l'algorithme, la compromission de celui-ci entrainte celle de toutes les communications qu'il l'utilise, alors qu'en changeant de clef, on limite l'effet d'une compromission de la clef en la changeant régulièrement. Comme corollaire, on peut entendre ce principe comme le fait que « la sécurité par l'obscurité » n'est pas pérenne.

Solution de l'exercice 14

— Pour la clef de 40 bits, c'est 2^{16} fois plus rapide : $2^{40}/2^{56} \times (4,5 \times 3600 \times 24) \approx 5,9s$
— Pour la clef de 56 bits, c'est 2^{56} fois plus lent : $2^{112}/2^{56} \times (4,5 \times 3600 \times 24) = 2^{56} \times (4,5 \times 3600 \times 24) \approx 2,8 \times 10^{22}s$
— En moyenne on a besoin de parcourir que la moitié de l'espace de recherche, c'est donc 9 jours dans le pire cas.

Solution de l'exercice 15

1. Il est nécessaire d'avoir une clef symétrique pour chaque couple de correspondants potentiels, soit $C_n^2 = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$. Pour le cas où on considère un sous-ensemble des membres et non plus seulement des couples de correspondants, on a besoin de 2^n clefs (on peut éventuellement retirer les singletons, le groupe vide et le groupe de tous, soit $n+2$ clefs).
2. Chaque membre doit posséder un couple de clef, il faut donc n couples pour le groupe
3. On résume les points faibles et forts dans le tableau suivant :

	Points forts	Points faibles
Symétrique	Rapide à calculer	Nécessite un canal sûr (confidentiel et authentique)
Asymétrique	Les clefs de chiffrements sont publiques : échanges sur canal non-sûr, possibilité de signer les messages	Calculs coûteux

Les systèmes symétriques ne permettent pas la signature car on ne sait pas de qui parmis ceux qui connaissent le secret qui a chiffré un message. Un système hybride permet de profiter des atouts de l'asymétrique et des performances du chiffrement symétrique : le chiffrement public peut permettre de s'authentifier mutuellement puis d'échanger une clef de session symétrique à usage unique.

Solution de l'exercice 16

1. On génère une paire RSA, on s'en sert pour créer un certificat où la clef publique et l'identité sont signés avec la clef privée. Ensuite on affiche le certificat. On est donc dans un cas où l'on se signe soit-même.
2. La clef publique RSA, *id est* les sections `Modulus` et `Exponent` du certificat.
3. C'est l'algorithme symétrique pour protéger la partie privée de la clef RSA. Toute information relative à cette partie n'a rien a faire dans le certificat. Le mot de passe est demandé trois fois : les deux premières fois à la génération de la paire RSA (une fois pour vérification), la

troisième c'est lorsqu'on se sert de cette clef pour générer le certificat car il faut accéder à la partie privée de la paire.

Solution de l'exercice 17

1. On a autant de clefs possibles que de symboles dans l'alphabet considéré.
2. On supprime les caractères spéciaux, la casse et les caractères diacritiques. On passe alors de `attaquezmaintenant` à `gzzgwakfsgotzktgtz`.
3. C'est le même clair.
4. On peut utiliser la fréquence relative des lettres de la langue pour déterminer le pas pour 'e' et ainsi déchiffrer tout le message car le pas est constant.
5. Le clair est `cestuneenigme` chiffré avec la clef 7.
6. Il y a désormais 26! clefs possibles, c'est beaucoup plus gros, mais les fréquences de lettre resteront préservées lors du chiffrement (un e sera toujours chiffré de la même façon) ce qui rend l'attaque plus compliquée mais toujours possible.

Solution de l'exercice 18

1. C'est indistinguable du cas où la clef est simplement `bonjour` : ça ne sert donc à rien de répéter un mot.
2. On obtient `ckrpjigqkpbbvvlpgh`
3. Si elles sont chiffrées par les même morceaux de clefs.
4. On aura une relation de multiples entre les longueurs et la longueur de la clef sera un sous-multiple des ppcm des longueurs identifiées. Voir `http://fr.wikipedia.org/wiki/Cryptanalyse_du_chiffre_de_Vigen%C3%A8re` pour plus de détails.
5. Dans `eiwemcgjratpdickwseiwemciiyeawg` on voit `eiwemc` apparaître deux fois avec un écart de 18. Le mot de passe est donc de longueur $2, 3, 6, 9$ ou 18.
6. Une fois la longueur n de la clef déterminée, on fera une analyse statistique en considérant n alphabets indépendants : un pour chaque lettre du secret.
7. C'est le *One-Time Pad* ou chiffrement jetable de Vernam. C'est inconditionnellement sûr (en un sens technique que l'on ne développera pas) mais très difficile voir impossible à mettre en œuvre, en effet il est très difficile d'obtenir de longs mots réellement aléatoire et la communication du secret est problématique.

Solution de l'exercice 19

1. $m^e \pmod n$ est ici $21^{103} \pmod{143} = 21^{25\cdot 4+3} \pmod{143} = 21^3 \cdot 21^{25\cdot 4} \pmod{143}$, on va ensuite éliminer le gros exposant avec l'indice donné : $21^{103} \pmod{143} = 21^3 \pmod{143} \cdot 21^{25\cdot 4} \pmod{143} = 21^3 \pmod{143} = 21 \cdot (21 \cdot 21 \pmod{143})$, on utilise enfin la division Euclidienne $21^{103} \pmod{143} = 21 \cdot (441 - 3 * 143) = 21 \cdot 12 = 109$
2. On a $n = 11 \cdot 13$ et on cherche d tel que $103 \cdot d \equiv 1 \pmod{(11-1)\cdot(13-1)}$. Comme $6 \cdot 120 = 720$ et $7 \cdot 103 = 721$ on a bien $7 \cdot 103 \equiv 1 \pmod{120}$ et la clef privée est $(7, 143)$.
3. Il y a d'une part le problème de la génération de *grands* nombres premiers p et q, et les calculs d'exponentielles qui sont trop couteux si effectués naïvement. Enfin, il y a des problèmes de canaux cachés : il faut que l'observation du processeur qui exécute ne puisse pas fournir d'information sur les secrets qui sont manipulés. Des attaques sur le *timing* des bits de sorties ou sur la consommation éléctrique existent, c'est pourquoi les algorithmes sont souvent randomisés.

Solution de l'exercice 20

« **je sais** » connaissance d'un mot de passe, d'un secret.

« **je possède** » possession d'un objet : carte à puce, badge. On peut éventuellement placer ici les « objets immatériels » comme les cookies.

« **je suis** » caractéristique biométrique : doigt, voix, iris.

Solution de l'exercice 21

I : Généralités

1. Comme le hachage est irréversible, le vol du fichier des hachés ne révèle pas les mots de passes.
2. Car en disposant du fichier des hachés on peut tenter une attaque au dictionnaire et révéler les plus faibles.
3. Ce serait inutile si les mots de passes choisis étaient suffisament sûrs.
4. — l'usage du nom d'utilisateur ou une modification simple de celui-ci ;
   — les mots de passes trop courts ;
   — l'utilisation de mots du dictionnaire, de noms propres, d'acronyme ou des concaténations de mots courts ;
   — des modifications triviales des derniers (suffixe, mixed case sur voyelles, mirroir etc.)
   — l'usage de nombres structurés (date, numéro de téléphone, etc.)
   — les séquences du clavier (azerty, 1793, aqzsed, aqwzsx)
   — ne pas faire varier la casse, ne pas utiliser de nombres
5. Il vaut mieux avoir une mnémotechnie s'appuyant sur une phrase à mémoriser puis modifiée selon des règles. Exemple « J'ai été diplômé en 2004 »
   1. « J'ai été » $\mapsto$ *G&t*
   2. « diplômé » $\mapsto$ *10Plm*
   3. « en 2004 » $\mapsto$ *2k4*

   Soit au final : *G&t10Plm2k4*. Ici le mot de passe et court et les règles sont assez compliquée mais on peut prendre une phrase plus longue avec des mots au hasard et des règles plus simples. Voir par exemple pour la blague `https://xkcd.com/936/` et `https://xkpasswd.net/s/` ou `http://www.diceware.com` pour des générateurs.

II : Dénombrement

1. Il y a $26+10$ caractères. On compte $\sum_{i=0}^{7} 36^i = \frac{1-36^8}{1-36} = 80603140213$ (environ 80 milliards) d'après la formule sur la somme des termes d'une suite géométrique.
2. Au rythme de 1867400 essais par seconde, il faut 4316 secondes, soit environ 1h12min pour tout explorer.

Solution de l'exercice 22

Supposons qu'un utilisateur C obtienne le premier message $K' = \{\{K\}_{K_{a^{-1}}}\}_{K_b}$. Il relance le protocole avec B :

$C \to B$: $\{\{K'\}_{K_{c^{-1}}}\}_{K_b}$

$B \to C$: $\{\{K'\}_{K_{b^{-1}}}\}_{K_c}$

Ainsi, C reçoit $\{\{K'\}_{K_{b^{-1}}}\}_{K_c} = \{\{\{\{K\}_{K_{a^{-1}}}\}_{K_b}\}_{K_{b^{-1}}}\}_{K_c}$. Comme, $\{\{m\}_{K_{x^{-1}}}\}_{K_x} = m$, C obtient $\{\{K\}_{K_{a^{-1}}}\}_{K_c}$ qu'il peut déchiffrer.

En fait, c'est un protocole qui ne garantit pas grand chose, voire rien...

Solution de l'exercice 23

1. K_{AB} est la clef secrète (symétrique) partagée entre A et B, K_{AS} est celle partagée entre A et le serveur S, K_{BS} est celle partagée entre B et le serveur S. Ce protocole permet l'établissement d'une clef de session K_{AB}.
2. Voici l'échange où I relance deux sessions en usurpant B puis A :
   1. $A \to S : A, \langle T_A, B, K_{AB} \rangle_{K_{AS}}$
   2. $S \to B : \langle T_S, A, K_{AB} \rangle_{K_{BS}}$
   3. $I(B) \to S : B, \langle T_S, A, K_{AB} \rangle_{K_{BS}}$
   4. $S \to A : \langle T'_S, B, K_{AB} \rangle_{K_{AS}}$
   5. $I(A) \to S : B, \langle T'_S, B, K_{AB} \rangle_{K_{AS}}$
   6. $S \to B : \langle T''_S, A, K_{AB} \rangle_{K_{BS}}$
3. L'attaquant peut ainsi mettre à jour la durée de validité d'une clef (potentiellement comprise) K_{AB} alors que les participants A et B croient qu'elle est périmée.

Solution de l'exercice 24

1. Avec modulo 11 on a dans cet ordre $\{2^0, 2^1, \ldots, 2^9\} = \{1, 2, 4, 8, 5, 10, 9, 7, 3, 6\}$ et $2^{10} = 1024 = 93 \times 11 + 1 = 1 \mod 11$ (le groupe est cyclique et fini). $g^a = 2^7 \mod 11 = 7$. $g^b = 2^5 \mod 11 = 10$. Pour générer en Haskell par exemple les 20 premiers nombres via le générateur `take 20 . map (\x -> 2^x `mod` 11) $ [0..19]`
2. Alice a connaissance de $(g^b)^a$ (mais aussi de a, p et de g) et Bob de $(g^a)^b$, or il s'agit du même nombre car $(g^b)^a = g^{b \times a} = g^{a \times b} = (g^a)^b$. Ce secret partagé sera la graine de la clef d'un chiffrement symétrique des communications ultérieures (avec AES ou DES par exemple). On peut par exemple choisir les n premiers bits de $h(g^{a \times b})$ où h est une fonction de hashage cryptographique pour constituer la clef symétrique.
3. C'est le problème du logarithme discret (DLP), pour lequel on ne connaît pas d'algorithme polynomial. Dans ce cas c'est assez facile car p est petit : $2^6 = 9 \mod 11$ et $2^8 = 3 \mod 11$, le secret est donc $2^{6 \times 8} = 3 \mod 11$.
4. Eve connaît p, g, G (supposés publics), g^a et g^b mais pas $g^{a \times b}$. La difficulté de découverte du secret réside dans la difficulté à calculer $g^{a \times b}$ connaissant p, g, G, g^a et g^b, c'est *le problème de Diffie-Hellman* (DHP). Si on sait résoudre le DLP efficacement, alors le DHP serait aussi résolu efficacement (car connaissant a et b il est facile de calculer $a \times b$) et le protocole serait cassé. L'assertion réciproque (résoudre efficacement le DHP implique une résolution efficace du DLP) est une question ouverte.
5. Diffie-Hellman est vulnérable à une attaque de type *Man In The Middle* (MITM) décrite comme suit :
   1. Eve tire deux nombres au hasard a' et b' ;
   2. Eve intercepte g^a est envoie $g^{a'}$ à Bob en se faisant passer pour Alice ;
   3. Eve intercepte g^b est envoie $g^{b'}$ à Alice en se faisant passer pour Bob.

   A l'issue de ce protocole, Alice connaît $g^{b'a}$, secret qu'elle croit partager avec Bob. Bob de son côté connaît $g^{ba'}$ qu'il croit partager avec Alice. Eve a connaissance de a' (tiré au hasard), b' (tiré au hasard), g^a (envoyé par Alice) et $g^a b$ (envoyé par Bob) : elle peut donc calculer $(g^a)^{b'}$ et $(g^b)^{a'}$. Dès lors, un message chiffré par Alice avec son secret $g^{b'a}$ sera déchiffré puis chiffré avec $g^{ba'}$ avant de le transmettre à Bob. Les deux parties auront donc l'impression de communiquer via une clef secrète partagée alors qu'il n'en est rien.
6. Chaque partie tire son propre nombre au hasard a, b et c, ensuite :

1. Alice envoie g^a à Bob
2. Bob envoie g^{ab} à Carole qui calcule g^{abc} le secret partagé
3. Bob envoie g^b à Carole
4. Carole envoie g^{bc} à Alice qui calcule g^{bca}
5. Carole envoie g^c à Alice
6. Alice envoie g^{ca} à Bob qui calcule g^{cab}

Il y a donc 6 messages dans ce cas.

Solution de l'exercice 25

1. diagramme de séquence ou listing en notation standard : $A \rightarrow TP : [A, \{B, K_{A,B}\}_{K_A}]$ suivit de $TP \rightarrow B : [\{A, K_{A,B}\}_{K_B}]$.
2. Car il ne connait pas K_A.
3. Car il ne connait pas K_B.
4. On peut rejouer un message chiffré avec $K_{A,B}$, imaginer le cas où le message clair est « payer 100 €à M » qui est renvoyé *ad lib*.
5. On peut ajouter un horodatage aux messages ou un numéro de séquence (on jete les messages dont le numéro n'est pas plus grand que le dernier).

Solution de l'exercice 26

Voir `http://www.lsv.ens-cachan.fr/Software/spore/andrew.html`

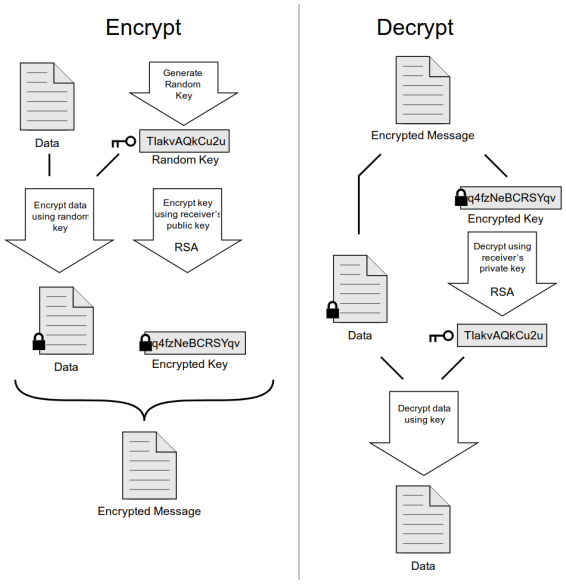
1. Respectivement : une clef partagée entre A et B, un nombre aléatoire généré par A, un nombre aléatoire généré par B.
2. Il sert à établir une nouvelle clef (de session) générée par B.
3. On joue une première fois le protocole puis on l'exécute une deuxième fois les trois premières étapes normalement en intervenant sur la dernière : l'attaquant intercepte le message destiné à A et lui en envoie le vieux.

    1. $A \rightarrow B : A, \{N'_a\}_{K_{ab}}$
    2. $B \rightarrow A : \{N'_a + 1, N'_b\}_{K_{ab}}$
    3. $A \rightarrow B : \{N'_b + 1\}_{K_{ab}}$
    4. $B \rightarrow I(A) : \{K''_{ab}, N''_b\}_{K_{ab}}$
    5. $I(B) \rightarrow A : \{K'_{ab}, N'_b\}_{K_{ab}}$

    A ne peut pas vérifier la fraîcheur de la clef, on peut donc le forcer à accepter une vieille clef, ici K'_{ab}, qui est possiblement compromise.

Solution de l'exercice 27

1. Voir l'illustration de Wikipedia ci-dessous



2. Il y a un chiffrement symétrique par une *passphrase*. Typiquement un `aes-256`.
3. Alice et Bob doivent s'échanger leurs clefs publiques de façon sûre. Ils peuvent par exemple s'échanger les clefs de la main à main ou plus simplement, s'échanger les clefs sur un canal non sûr puis vérifier l'intégrité des clefs transmises en se donnant mutuellement les hachés des fichiers reçus.
4. C'est normal, car Alice n'a pas besoin d'accéder à sa clef privée qui est protégée par le mot de passe pour chiffrer un message à Bob. S'eut été nécessaire si Alice avait *signé* le message.
5. PGP n'utilise pas directement la clef publique du destinataire pour chiffrer un message : c'est un système hybride qui génère une clef de session symétrique pour chaque message. C'est cette clef qui est chiffrée avec la clef publique du destinataire. Dans le cas où il y a plusieurs destinataires, il y a une seule clef symétrique, chiffrée indépendamment pour chaque destinataire.
6. Il faut également qu'elle sauvegarde sa clef privée, sans quoi sa sauvegarde sera inutilisable.

Solution de l'exercice 28

1. L'utilisateur n'a pas à s'authentifier à chaque fois grâce au mécanisme de ticket : une fois le TGT obtenu, C peut le présenter plusieurs fois dans la période de validité.
2. — Seul l'AS a besoin de connaître le secret de l'utilisateur et ce secret n'a à transiter qu'une fois pendant la période de validité du ticket ;
   — Le problème réside dans le fait que si un ticket est compromis (e.g., la clef de session est compromise), il n'y a pas de révocation possible et ce dernier est utilisable tant que la période de validité n'est pas échouée.
3. Effectivement, c'est un problème : le vol de l'information partagée permet à un pirate de se faire passer pour C. Comme pour Needham-Schroeder qui est à la base de Kerberos, il existe une version asymétrique du protocole qui contourne cette faiblesse. Voir la `http://www.ietf.org/rfc/rfc4556.txt`
4. Le C envoie $\{A_C\}_{K_{C,S}}$ à S, l'authentificateur contient l'identité de C et une estampille temporelle. Un client qui crée un authentificateur valide est bien en possession de $K_{C,S}$, or cette clef a été envoyée par le TGS à C, chiffrée avec $K_{C,TGS}$ mais aussi à C via un ticket chiffré avec K_S. La clef K_S est connue seulement de S et TGS, comme C est authentifié auprès de TGS, S est sûr de bien avoir a faire à C.

Solution de l'exercice 29

1. La réponse est 42. On a $106 = f(1) = a_0 + a_1$ et $44 = f(4) = a_0 + 4a_1$. On résout le système. On a d'une part $f(4) - f(1) = 44 - 106 = 171 - 106 = 65$ et $f(4) - f(1) = 3a_1$, on en déduit que $a_1 = 65 \div 3 = 64$. On conclu que le secret $a_0 = f(1) - a_1 = 106 - 64 = 42$.
2. En général, le chiffrement symétrique est plus rapide que l'asymétrique. De plus, le coprocesseur peut faire de l'AES efficace mais *a priori* pas du RSA efficace.
3. $C_A = \{\langle A, K_A^{-1}, S, d\rangle\}_{K_C}$ avec $S \in \{PS, PA\}$ selon le type du participant et d une date d'expiration.
4. La confidentialité car doutes les données sont chiffrées et qu'une collusion pour obtenir M_A est supposée difficile. La disponibilité en partie car le mécanisme de sauvegarde permet de tout récupérer en cas de perte. Pour l'intégrité, on a pas d'information, mais on imagine que le *token* pourrait aussi vérifier l'intégrité des archives stockée dans le cloud.
5. Le patient A désire s'authentifier avec le professionnel B.
   1. $A \to B : \langle C_A, \{\langle A, n\rangle\}_{K_A}\rangle$ avec n une ombre aléatoire.
   2. $B \to A : \langle C_B, \{\langle B, PS, n\rangle\}_{K_B}\rangle$.
6. On prend un premier schéma avec $k = 2$ et $n = 2$ avec $S = M_A$ le secret à partager, on a deux morceaux $(i, f(i))$ et $(j, f(j))$. On distribue $(i, f(i))$ à tous les médecins de confiance. Ensuite, on prend un nouveau schéma avec $k = 2$ et n arbitraire, mais où $S = (j, f(j))$. On distribue les pièces aux différentes personnes de confiance. Ainsi, pour arriver à recouvrer M_A il faut le concours d'un médecin pour avoir $(i, f(i))$ et d'au moins deux personnes pour avoir $(j, f(j))$.
7. La patient A doit contacter un médecin pour obtenir et $(i, f(i))$ deux personnes pour avoir $(j, f(j))$. Ici, l'authentification est réalisée sur un canal auxiliaire, comme la rencontre des personnes physiques. À partir de ces données, il peut retrouver M_A, télécharger son dossier dans le cloud (qui comprendra entre autre $\{K_A\}_{M_A}$ et C_A), le déchiffrer et enfin importer le tout dans un *token* neuf.
8. $127 = 2^7 - 1$, on peut donc stocker des clefs de 6 bits (ou de 7 bits si on en brûle une). Pour la clef AES, il faut trouver le plus petit nombre premier supérieur à 2^{256}.
9. Avec h une fonction de hachage, on peut avoir par exemple $M_A^n = h^n(M_A)$.