Solutionnaire - Examen Formatif – S3i – APP6 : Sécurité des systèmes informatiques

Professeur : Frédéric Mailhot

1. **Pour les clés RSA, on choisit e et d de telle sorte que $d = e^{-1} \bmod \phi(n)$.**

   a) **Expliquez pourquoi.**
   Parce qu'ainsi, $d \times e = 1 \bmod \phi(n) = k \times \phi(n) + 1$. Or $c = m^e \bmod n$ et on désire que $m = c^d \bmod n$. Ceci implique que $m = m^{e \times d} \bmod n = m^{k \times \phi(n)+1} \bmod n$. Puisque $m^{\phi(n)} = 1 \bmod n$ (théorème d'Euler), alors $m^{k \times \phi(n)} = 1 \bmod n$, et donc $m^{e \times d} = m \bmod n = m$. On réobtient ainsi le nombre de départ.

   b) **Expliquez ce qui arrive si e et $\phi(n)$ ne sont pas relativement premiers.**
   Lorsque e et $\phi(n)$ ne sont pas relativement premiers, alors il n'existe pas d'inverse multiplicatif de e modulo $\phi(n)$, puisque $PGCD(e, \phi(n)) \neq 1$, et qu'il n'existe aucun nombre qui pourra multiplier e pour donner 1.

   c) **Comment peut-on faire pour prédéterminer la valeur de e, avant même d'avoir trouvé n et $\phi(n)$?**
   Il s'agit de choisir les facteurs p et q de n de telle sorte que (p – 1) et (q – 1) ne soient pas des multiples de la valeur de e choisie. De cette façon, $\phi(n)$ ne sera pas un multiple de e, et donc il est certain que e aura un inverse multiplicatif modulo $\phi(n)$.

   d) **Pourquoi devrait-on choisir une petite valeur pour e?**
   Pour simplifier les calculs de tous ceux qui voudront utiliser notre clé publique pour nous envoyer un message sécurisé. En effet, une petite valeur de l'exposant se traduit par un calcul beaucoup plus rapide par la méthode binaire des exposants.

2. **Le théorème du reste chinois (TRC) est utilisé pour accélérer les calculs de la méthode RSA.**
   a) **Trouvez la représentation TRC de m = 18 avec la base n = p X q = 5 X 7 = 35.**
   m = (18 mod 5, 18 mod 7) = (3, 4)
   b) **En supposant que p et q sont habituellement de tailles comparables (même nombre de bits), que pouvez-vous dire de la taille d'un message m comparé à la taille de sa représentation TRC? Supposez que m utilise le même nombre de bits que n.**
   En supposant que n et m sont de taille comparable, cela implique que m utilise environ 2 fois plus de bits que p ou q. En effectuant le modulo p et le modulo q de m, alors les résidus respectifs ne pourront dépasser p ou q. Ainsi, le message m sera maintenant représenté par deux nombres de taille comparable à celle des nombres p et q. m sera donc devenu une liste de deux nombres utilisant chacun deux fois moins de bits que le message initial. On remarque toutefois que la quantité d'information est la même : le nombre total de bits est toujours comparable à celui du nombre initial.
   c) **Soient deux nombres de 32 bits x et y, et deux nombres de 64 bits z et t. Le calcul du produit de z par t prendra 4 fois plus de temps que le calcul du produit de x par y. En quoi cela favorise-t-il l'utilisation du TRC?**
   Puisque m sera formé d'une liste de deux nombres deux fois plus petits, qu'on pourra utiliser indépendamment lors d'opérations arithmétiques, le temps de calcul

sur chacun des deux nombres sera 4 fois plus rapide. Puisqu'il y a deux de ces nombres, nos calculs seront donc 4 / 2 = 2 fois plus rapides si on utilise les résidus TRC au lieu du message original complet.

3. **La méthode de Miller-Rabin est utilisée pour déterminer si un nombre est premier**
   a) **Pourquoi cette méthode est-elle préférable à celle du petit théorème de Fermat?**

   Parce que la méthode basée sur le théorème de Fermat peut difficilement détecter que les nombres de Carmichael ne sont pas premiers. En effet, seuls les multiples des facteurs premiers d'un nombre de Carmichael n'obéiront pas à l'équation $a^{(p-1)} = 1 \bmod p$.

   b) **Est-on certain qu'un nombre identifié comme premier par la méthode de Miller-Rabin est effectivement premier? Pourquoi?**

   Non, la méthode de Miller-Rabin est une méthode probabiliste. À chaque itération, il y a un probabilité ¼ qu'on indique qu'un nombre est premier alors qu'il ne l'est pas. Par contre, en utilisant un nombre suffisant d'itérations, on peut réduire cette probabilité autant qu'on le désire.

   c) **Expliquer pourquoi on divise itérativement (p – 1) par 2 dans cette méthode. Pourrait-on diviser par un autre nombre premier? Expliquer pourquoi.**

   La raison en est que la racine carrée de 1 ne peut qu'être 1 ou p – 1 lorsque p est premier. En trouvant le plus grand facteur impair de (p – 1), qu'on utilise par la suite pour élever un nombre « a » à cette puissance, on permet le calcul de puissances de 2, ce qui nous permet d'utiliser la racine carrée (la puissance précédente).

   Utiliser un nombre premier autre que 2 n'aurait pas beaucoup de mérite : on ne sait pas nécessairement quelles sont les valeurs acceptables de la racine 3^e ou 5^e de 1 modulo p, alors que la racine carrée de 1 modulo n, elle, est bien connue.

4. **La méthode de Diffie-Hellman a été la première méthode à clé publique connue de tous. Est-ce que cette méthode aurait été utilisable sans l'existence de méthodes d'encryption symétrique? Expliquez votre réponse.**
   Non. La méthode de Diffie-Hellman permet la création d'un nombre secret, connu seulement des deux interlocuteurs (Alice et Bob). Mais pour que ce nombre secret soit utile, on doit s'appuyer sur une autre méthode pour encrypter les échanges subséquents. DH repose donc sur l'utilisation subséquente d'une technique d'encryption symétrique telle DES, 3DES, AES ou autre, où la clé d'encryption est justement le nombre secret connu seulement d'Alice et de Bob.

5. **Expliquez pourquoi la méthode rho de Pollard utilise l'algorithme de Floyd. En effet, cette méthode requiert 3 fois plus d'appels à la procédure de calcul du polynôme f(x) que la méthode de Pollard initiale**.
   Si on n'utilise pas l'algorithme de Floyd, on risque d'utiliser une très grande quantité de mémoire pour conserver les valeurs intermédiaires. Comme il arrive souvent en informatique, il s'agit d'un compromis entre la quantité de mémoire requise et le temps d'exécution. Ici, le temps d'exécution est multiplié par trois,

mais la mémoire requise passe d'une quantité potentiellement très grande à une constante (2).

6. **Soit un échange de messages numériques signés entre Alice et Bob.**

a) Expliquez comment Alice peut utiliser un système à clé publique tel RSA pour signer un document électronique destiné à Bob.

Alice utilise sa clé privée pour chiffrer le message destiné à Bob. Ainsi Bob, en déchiffrant le message à l'aide de la clé publique d'Alice, pourra savoir qu'elle seule a pu ainsi encrypter le message.

b) Maintenant Ève entre en scène : Alors qu'Alice envoie un message signé à Bob, Ève intercepte le message d'Alice, lui substitue son propre message signé, et fait croire à Bob qu'il est signé par Alice. Expliquez comment Ève a pu procéder pour faire accepter à Bob que sa signature est bien celle d'Alice.

Si Ève a envoyé une clé publique à Bob en lui faisant croire que cette clé est la clé publique de Alice.

c) Supposons qu'Alice envoie à Bob des messages encryptés à l'aide de la clé publique de celui-ci. Comment Ève pourrait-elle procéder pour pouvoir lire les messages destinés à Bob? Dans ce cas, si le même message parvient à Bob, va-t-il s'en apercevoir? Est-ce que Ève peut faire quelque chose pour cacher son jeu?

Éve va devoir envoyer deux clés publiques de son cru : l'une à Alice (en lui faisant croire que c'est la clé publique de Bob) et l'autre à Bob (en lui faisant croire que c'est la clé publique de Alice). Lorsqu'elle reçoit le message d'Alice, Ève sera en mesure de le déchiffrer, en utilisant la clé privée qui correspond à la clé publique envoyée à Alice. Puis, en utilisant la clé privée correspondant à la clé publique qu'elle a envoyée à Bob, Ève peut 'signer' le message à la place d'Alice. Enfin, Ève encrypte le message destiné à Bob à l'aide de la vraie clé publique de celui-ci.

7. **Soit un programme dans lequel existe une vulnérabilité au problème de « buffer overflow ».**

a) Si le problème est dû à strcpy(), croyez-vous qu'il serait possible de modifier cette procédure système pour éliminer le problème? Si oui, comment? Si non, quelle information supplémentaire devrait-on posséder? Est-il possible d'avoir accès à cette information?

Sans connaître la taille du tampon dans lequel on fait la copie, il est impossible de modifier strcpy() pour s'assurer qu'il n'y ait pas de dépassement. Par contre, si toutes les allocations de mémoire sont enregistrées, il devient possible de vérifier la taille du tampon qui débute à une adresse donnée. On pourrait ainsi modifier strcpy() et toujours vérifier la taille du tampon passé en paramètre avant d'y faire une copie.

b) Si toutes les procédures systèmes étaient modifiées pour enrayer leur susceptibilité au problème de « buffer overflow », est-ce que les applications bâties à l'aide de ces procédures systèmes seraient alors sécuritaires?

Pas nécessairement. Il se pourrait qu'un développeur utilise sa propre boucle (while ou for), sans vérifier la taille des tampons dans lesquels il y a des copies de faites.

c) Selon vous, serait-il possible de recompiler la même application (sans modification de code source) à l'aide d'un compilateur différent, et d'éliminer le problème de « buffer overflow »? Si, oui, expliquez comment. Si non, expliquez pourquoi.

Plusieurs solutions sont possibles. Il s'agit soit d'empêcher que le débordement d'une variable vienne réécrire les adresses de retour, soit de détecter que c'est effectivement arrivé, soit d'empêcher l'exécution de code situé dans le stack. Pour la première solution, on peut imaginer l'utilisation de stacks multiples. On met alors les adresses de retour sur le stack1, et les variables automatiques sur le stack 2. Pour la deuxième solution, on utilise ce qu'on appelle un canari (ce terme fait référence aux canaris qui étaient utilisés jadis pour détecter les fuites de grisou dans les mines de charbon). Sous l'adresse de retour dans le stack, on réserve quelques octets dans lesquels on inscrit une valeur connue du programme (possiblement variable d'une exécution à une autre), et qu'on vérifie avant d'effectuer le retour de la procédure. Si cette valeur a été modifiée, c'est que le canari s'est évanoui, et on interrompt l'exécution du programme. La troisième solution implique de protéger l'espace mémoire du stack en exécution. S'il n'est accessible qu'en lecture ou en écriture, le système détectera automatiquement une tentative d'exécution de code dans le stack, et il y aura immédiatement un 'segmentation fault'.

8. **Vous êtes responsable de la sécurité dans une entreprise de moyenne envergure, où les usagers des systèmes informatiques sont peu connaissant des vulnérabilités liées aux mots de passe. Donnez quelques règles simples pour les aider dans le choix d'un mot de passe sécuritaire.**

Les mots de passe doivent être assez long (8 caractères ou plus), et ne doivent correspondre à aucun mot existant. De plus, l'utilisation de chiffres et de signes de ponctuation augmente de beaucoup la sécurité des mots de passe. Une méthode simple est d'utiliser une phrase dont on ne prend que la première lettre de chaque mot, le tout parsemé de chiffres et de signes de ponctuation. On peut aussi simplement utiliser une phrase très longue, qui utilise des mots peu communs. Si le système permet des mots de passe de longueur très grande, c'est tout aussi sécuritaire que d'utiliser plus de caractères dans un mot de passe plus court.