Sécurité informatique et cryptographie

GUIDE DE L'ÉTUDIANTE ET DE L'ÉTUDIANT
S3 Génie Informatique – APP6

Été 2023 – Semaines 12, 13 et 14

Département de génie électrique et de génie informatique
Faculté de génie
Université de Sherbrooke

Historique des modifications

Date	Responsables	Description
22 mars 2004	Jean-Denis Hatier, Alain Houle, Frédéric Mailhot	Version 0.1
16 juillet 2004	Frédéric Mailhot	Version 1.0
26 juillet 2004	Frédéric Mailhot	Version 1.1
8 octobre 2004	Frédéric Mailhot	Version 2.0
30 mars 2005	Frédéric Mailhot	Version 3.0
14 juillet 2005	Frédéric Mailhot	Version 3.1
14 novembre 2005	Frédéric Mailhot	Version 3.2
2 juillet 2006	Frédéric Mailhot	Version 3.3
juin 2008	Frédéric Mailhot	Version 4.0
juillet 2009	Frédéric Mailhot	Version 5.0
juillet 2010	Frédéric Mailhot	Version 5.1
juillet 2011	Frédéric Mailhot	Version 5.2
juillet 2012	Frédéric Mailhot	Version 5.3
juillet 2013	Frédéric Mailhot	Version 5.4
juillet 2014	Frédéric Mailhot	Version 5.5
juillet 2015	Frédéric Mailhot	Version 5.6
juillet 2016	Frédéric Mailhot	Version 5.7
juin 2018	Frédéric Mailhot	Version 5.8
novembre 2018	Frédéric Mailhot	Version 6.0
juin 2019	Frédéric Mailhot	Version 6.1
octobre 2019	Frédéric Mailhot	Version 6.2
juin 2020	Frédéric Mailhot	Version 6.3
novembre 2020	Frédéric Mailhot	Version 6.4
juin 2021	Frédéric Mailhot	Version 6.5
juin 2022	Frédéric Mailhot	Version 6.6

Auteur : Frédéric Mailhot
Version : 6.7 (Juillet 2023)

Ce document est réalisé avec l'aide de LaTeX et de la classe `gegi-app-guide`.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

1 ÉNONCÉ DE LA PROBLÉMATIQUE

Les salaires sont connus de tous !

Une petite polémique a éclaté dans l'entreprise sherbrookoise Mégamillions. Depuis quelques jours, plusieurs employés ont défilé au bureau des ressources humaines, se plaignant que leur salaire est moindre que le salaire de certains autres employés. Les employés n'étant pas syndiqués, leur salaire est négocié au cas par cas et constitue une information confidentielle. Le directeur des ressources humaines se questionne donc sur la façon dont certains employés ont réussi à connaître le salaire des autres employés. Il émet l'hypothèse que le système informatique de l'entreprise souffre peut-être de lacunes de sécurité, ce qui aurait permis à quelques petits "génies" internes de mettre la main sur la liste de salaires. Vous obtenez le mandat de vérifier l'hypothèse du directeur des ressources humaines. Pour réaliser votre mandat, vous devez déterminer de quelle façon l'information confidentielle a été obtenue, et proposer des solutions pour empêcher que cette situation ne se reproduise.

Lors de votre visite de l'entreprise, vous apprenez que la liste des salaires est conservée dans le fichier "salaires.mm", de même qu'un fichier contenant de l'information sur le chiffrement, appelé "parametres.mm". Ces fichiers se trouvent sur une machine extérieure (louée de l'université de Sherbrooke), à laquelle seul le comptable peut accéder par *scp*, avec le *login* : "comptabl". La machine s'appelle "ssh.gegi.usherbrooke.ca", l'information intéressante se trouvant sous le répertoire "pub". L'option *-r* de *scp* pourrait s'avérer intéressante. Bernard Petit, le comptable, vous mentionne que dans un premier temps il a utilisé une clé publique RSA pour chiffrer les salaires apparaissant dans le fichier. Il vous confirme que l'entreprise utilise le système de clé RSA depuis longtemps, entre autres pour un système de boîte à suggestion. Comme lui et son patron sont les seuls à détenir la clé privée correspondante, il a décidé de réutiliser la même clé. Évidemment, tous dans l'entreprise connaissent cette clé publique, qui est :

$$(n = 860623810257574886804969187380595545083155447 97, e = 13)$$

M. Petit vous indique qu'il a consulté un expert en sécurité informatique, qui lui a conseillé d'utiliser une méthode de chiffrement symétrique, par bloc (méthode de type Feistel ou autre) ou par flux, pour protéger le fichier de salaires. Pour le chiffrement par bloc, l'expert a recommandé une clé d'au moins 128 bits, en utilisant une méthode comme AES. L'expert a aussi indiqué que si le fichier devait être échangé entre M. Petit et son patron, l'utilisation de la méthode Diffie-Hellman pour générer une clé symétrique serait appropriée. Il faudrait alors trouver un triplet de nombres (q, p, g).

Suite à une demande de son patron, M. Petit a effectivement utilisé la méthode Diffie-Hellman pour obtenir une clé symétrique. Cette dernière, quoique un peu courte, a été utilisée dans un système de chiffrement par flux (M. Petit a répété la clé autant de fois que nécessaire) pour chiffrer le fichier de salaire, chaque bloc de 8 bits de la clé étant utilisé pour chiffrer un caractère différent du fichier de salaire. C'est finalement ce fichier, "salaires.mm", qui est demeuré sur le serveur externe. Enfin, pour s'assurer que les paramètres Diffie-Hellman soient en sécurité, M. Petit les a chiffrés à l'aide de la méthode RSA, en utilisant une deuxième paire de nombres (n, e). Les paramètres Diffie-Hellman chiffrés, ainsi que la paire de nombres (n, e), se trouvent aussi sur la machine "ssh.gegi.usherbrooke.ca", dans le fichier "parametres.mm".

Lors d'une rencontre subséquente avec l'expert en sécurité, ce dernier a mentionné que le système mis en place par M. Petit était à la fois trop complexe et potentiellement mal configuré. L'expert a indiqué que le choix des nombres premiers était primordial, tant pour la méthode RSA que pour la méthode Diffie-Hellman. En particulier, il a mentionné que si la méthode de Fermat avait été utilisée au lieu de la méthode de Miller-Rabin pour vérifier la primalité des facteurs du nombre n, il existait une très faible possibilité que l'un d'eux soit un nombre de Carmichael. Dans ce cas, l'algorithme ρ (rho) de Pollard pourrait être utilisé pour factoriser n. Ensuite, à l'aide de la fonction indicatrice d'Euler $\phi(n)$ ("phi de n") et l'algorithme étendu d'Euclide, il serait possible de trouver la clé privée. L'expert a aussi indiqué que pour certains types de nombres premiers, l'algorithme "$p - 1$" ("pé" moins un) de Pollard pouvait être utilisé pour factoriser n. Enfin l'expert a indiqué que les paramètres (q, p, g) du système Diffie-Hellman devraient absolument répondre aux quatre critères de sécurité reconnus, sinon il était possible qu'il n'y ait pas assez de puissances distinctes de g. M. Petit vous indique que personne dans l'entreprise n'avait compris ce charabia, et donc qu'il s'en était tenu au système en place.

En furetant sur le serveur Linux de l'entreprise, vous remarquez qu'il y a plusieurs utilitaires très intéressants, qui sont disponibles pour l'ensemble des employés. Parmi ces utilitaires, vous remarquez le programme *wireshark*, qui est très utile pour comprendre et optimiser le trafic sur le réseau. Vous remarquez aussi qu'on a récemment ajouté un nouvel utilitaire, "infotel", appartenant à *root*, mais qui peut être exécuté par tous via la commande *sudo*. Ce programme a été développé par un consultant externe, qui a été très apprécié pour son travail particulièrement rapide. Le programme "infotel" prend un nom d'usager en entrée et donne son numéro de téléphone en sortie. Le réceptionniste de l'entreprise vous mentionne que son travail a été facilité depuis que l'utilitaire "infotel" est disponible. Il vous dit qu'il n'a jamais eu de problème, sauf la fois où il s'était appuyé par inadvertance sur une touche de clavier, et qu'il avait alors utilisé "infotel" sur un très long nom bidon. Le système avait

indiqué : "*segmentation fault*", et le réceptionniste avait dû recommencer. Le consultant avait été averti de ce problème mineur, mais il avait dit que c'était simplement un dépassement de tampon (buffer overflow) et qu'en utilisant des noms de vrais employés il n'y aurait jamais de problème. Le code source du programme, "infotel.c", se trouve encore dans le compte du consultant. On vous dit que le consultant s'occupe d'un nouveau mandat pour l'entreprise. Il travaille à distance, et communique fréquemment avec le serveur de l'entreprise en utilisant *telnet* et *ftp*.

En poursuivant votre enquête, vous apprenez que tous les employés ont un compte sur le serveur Linux de l'entreprise. Ce serveur utilise un système de mots de passe très bien protégé, avec à la fois un fichier public */etc/passwd* et un fichier caché */etc/shadow*, ce dernier n'étant lisible que de l'administrateur du système (*root*). Les mots de passes sont chiffrés avec la commande système *crypt()*, en utilisant l'extension *1* pour mieux protéger le tout. Vous savez fort bien que même si quelqu'un avait réussi à obtenir une copie du fichier */etc/shadow*, il leur aurait fallu aussi créer un programme utilisant *crypt()* qui aurait tenté de deviner les mots de passe (ou qui aurait fait une attaque de force brute en énumérant tous les mots de passe jusqu'à une certaine taille). Évidemment le travail de l'attaquant aurait pu être simplifié si aucun *SALT* n'avait été utilisé, permettant l'utilisation de *"Rainbow Tables"*.

Notes :

- Sur le serveur Linux de l'entreprise, vous remarquez que la commande *man* est présente. Vous savez qu'au besoin, *man* est très utile pour connaître les détails de la plupart des commandes et structures Linux. Par exemple, les commandes suivantes pourraient s'avérer intéressantes : *man man*, *man passwd*, *man shadow*, *man sudo*, *man crypt*, *man bc*, *man wireshark*, *man ftp*, *man sftp*, *man telnet*, *man ssh*, *man emacs*, *man vim*, *man sed*, *man perl*, *man python*, etc.
- Cette problématique est structurée autour de problèmes de sécurité sur un système Linux, mais les différents types de problèmes rencontrés ne sont pas spécifiques à ce système d'exploitation. En utilisant des outils apparentés à ceux utilisés ici, il aurait été possible de faire ressortir des classes de problèmes similaires sur d'autres systèmes d'exploitation tels Windows, UNIX ou macOS.

2 GUIDE DE LECTURE

2.1 Séquence d'étude suggérée

2.1.1 Lectures en lien avec la cryptographie (RSA, Diffie-Hellman, AES)

Introduction à la cryptographie :

Chiffrement par flux : section 1.2

Théorie des nombres : chapitre 2 (au complet)

Méthode ρ de Pollard : section 3.3.1

Méthode $p-1$ de Pollard : section 3.3.2

Diffie-Hellman et RSA : sections 4.1 et 4.3

Courte animation de l'exécution de AES

Si désiré : lien vers le logiciel open source Cryptool, un système Windows utilisé pour produire le vidéo précédent

Description de AES sur wikipédia (version anglaise, plus complète que celle en français)

Introduction à la cryptographie

Infrastructure de clés publiques

Le principe de Kerckhoffs

Fonctionnement de *crypt()* - **Lire attentivement la Note au sujet de glibc**

Gestion des mots de passe

Introduction à la cryptographie, DES, RSA : section 1.1 (pp 1-2)

Chiffrement symétrique : section 1.5 (pp 15-21)

2.1.2 Lectures en lien avec les problèmes de dépassement de tampon (buffer overflow)

Explication du problème de buffer overflow

Explication du problème de buffer overflow sur wikipedia (les versions anglaises et françaises sont différentes, mais se complètent).

Explication sur wikipedia de l'ordre des octets (*bytes*) dans la mémoire d'un ordinateur ("*endianness*") : utile pour comprendre comment représenter une adresse qui sera écrasée par un dépassement de tampon

2.1.3 Lectures en lien avec la commande Unix sudo

Courte explication de sudo

2.2 Lectures supplémentaires (non essentielles, mais utiles)

Petite présentation intéressante sur sudo

Site sur les nombres premiers

3 LOGICIELS ET MATÉRIEL

— L'environnement de travail sera une machine virtuelle Linux utilisant le logiciel VMWare (Workstation Pro (Windows ou Linux) ou Fusion (macOS)). Le fichier iso contenant la machine virtuelle est disponible sur le site de l'APP.

  — https://www.vmware.com/ca-fr/products/workstation-pro.html
  — https://www.vmware.com/ca-fr/products/fusion.html

  Vous pouvez utiliser la version d'évaluation. Pour prolonger, voir sur le site web pour des clés bonnes pour un an.

4 SANTÉ ET SÉCURITÉ

Dispositions particulières

Aucune.

5 SOMMAIRE DES ACTIVITÉS

Semaine 1

— Première rencontre de tutorat
— Étude personnelle
— Séminaire sur la théorie des groupes
— Formation à la pratique procédurale 1
— Formation à la pratique en laboratoire

Semaine 2

— Formation à la pratique procédurale/laboratoire
— Consultation facultative
— Étude personnelle et exercices
— Validation pratique de la solution

Semaine 3

— Rédaction du rapport d'APP
— Remise des livrables d'APP
— Deuxième rencontre de tutorat
— Évaluation formative théorique
— Évaluation sommative théorique

6 PRODUCTIONS À REMETTRE

— Les productions se font par équipe de 2, sauf lorsque indiqué autrement.
— L'identification des membres des équipes doit être faite sur la page web de l'unité avant 16h30, le lendemain de votre premier tutorat. Les personnes qui n'ont pas d'équipe d'APP à ce moment seront mis en équipe par le tuteur.
— La date limite pour le dépôt électronique est le jour de votre deuxième tutorat avant le début du premier groupe. Les retards seront pénalisés.
— Les productions soumises à l'évaluation doivent être originales pour chaque équipe, sinon l'évaluation sera pénalisée en cas de non-respect de cette consigne.

Rapport d'APP

Le rapport, nommé CIP1_CIP2.pdf, doit contenir les éléments suivants :

— Identification de l'ensemble des membres de l'équipe (Prénom, nom et CIP)
— Mot de passe du comptable, mot de passe du consultant
— Facteurs du nombre n contenu dans l'énoncé de la problématique, méthode utilisée
— Facteurs du nombre n utilisé pour chiffrer avec RSA les paramètres q, p et g, méthode utilisée
— Noms et salaires des employés de l'entreprise
— Le tout sous forme de liste, sur une page (deux au maximum)

7 ÉVALUATIONS

7.1 Grille d'évaluation

La note attribuée aux activités pédagogiques de l'unité est une note individuelle. L'évaluation portera sur les compétences figurant dans la description des activités pédagogiques. Ces compétences, ainsi que la pondération de chacune d'entre elles dans l'évaluation de cette unité, sont :

Activités et éléments de compétence		*Validation d'APP*	*Rapport d'APP*	*Examen sommatif*	*Examen final*
1	Compétence 1	45	15	110	140
1	Compétence 2	45	15	100	130
	Total : GIF380	90	30	210	270

Compétence 1 : Mettre en oeuvre une technique de chiffrement appropriée répondant à des critères spécifiques de sécurité.

Compétence 2 : Analyser les failles de sécurité dans un système informatique et proposer des solutions appropriées.

7.2 Qualités de l'ingénieur

La grille d'indicateurs utilisée aux fins de l'évaluation est donnée au tableau 7.1. Il est à noter qu'un niveau d'atteinte d'un indicateur dans cette grille n'a pas la même signification qu'un niveau d'atteinte d'une qualité dans le tableau **??**. Cela est normal, un indicateur et une qualité, ce sont deux choses différentes.

Tableau 7.1 Grille d'indicateurs utilisée pour les évaluations

Indicateur	Qualité	Aucun (N0)	Insuffisant (N1)	Seuil (N2)	Cible (N3)	Excellent (N4)
Démontrer, à un niveau universitaire, l'acquisition de connaissances en mathématiques	Q01.1	Ne résout pas ou très peu de problèmes mathématiques en génie	Résout correctement peu de problèmes mathématiques en génie	Résout correctement certains des problèmes mathématiques en génie	Résout aisément les problèmes mathématiques en génie	Résout aisément et efficacement les problèmes mathématiques en génie
Démontrer, à un niveau universitaire, l'acquisition de connaissances en sciences naturelles	Q01.2	N'applique pas ou très peu de concepts fondamentaux en sciences naturelles	Applique correctement peu de concepts fondamentaux en sciences naturelles	Est capable d'appliquer correctement certains des concepts fondamentaux en sciences naturelles	Applique aisément les concepts fondamentaux en sciences naturelles	Applique aisément et efficacement les concepts fondamentaux en sciences naturelles
Démontrer, à un niveau universitaire, l'acquisition de connaissances en sciences du génie	Q01.3	N'applique pas ou très peu de concepts fondamentaux en sciences du génie	Applique correctement peu de concepts fondamentaux en sciences du génie	Est capable d'appliquer correctement certains des concepts fondamentaux en sciences du génie	Applique aisément les concepts fondamentaux en sciences du génie	Applique aisément et efficacement les concepts fondamentaux en sciences du génie
Élaborer une procédure de résolution	Q02.2	Élabore une procédure de résolution de problème incomplète ou inappropriée	Élabore une procédure de résolution de problème minimalement appropriée	Élabore une procédure convenant à la résolution du problème, sans être parmi les meilleures	Élabore une procédure générale de résolution appropriée	Élabore une procédure ingénieuse de résolution
Appliquer la procédure de résolution	Q02.3	Est incapable de mettre en uvre la procédure de résolution choisie	Commet des erreurs mineures dans l'application de la procédure de résolution choisie	Commet peu d'erreurs mineures dans l'application de la procédure de résolution choisie	Applique correctement la procédure de résolution choisie	Applique efficacement et rigoureusement la procédure de résolution choisie
Analyser et interpréter les résultats obtenus	Q02.4	Est incapable d'analyser les résultats obtenus et d'en identifier les limites et la portée	Est capable d'analyser partiellement les résultats obtenus et d'en identifier les limites et la portée	Est capable d'analyser sommairement les résultats obtenus et d'en identifier les limites et la portée	Est capable d'analyser correctement les résultats obtenus et d'en identifier les limites et la portée	Est capable d'analyser correctement les résultats obtenus et témoigne d'une compréhension fine de leurs limites et de leur portée
Rechercher plusieurs solutions et en sélectionner une	Q04.3	Identifie peu de solutions et éprouve de la difficulté à sélectionner celle qui semble convenir	Identifie quelques solutions et sélectionne celle qui semble convenir, sans toutefois en faire la validation	Identifie plusieurs solutions, fait une analyse sommaire et sélectionne celle qui semble être la meilleure. Valide sommairement le potentiel de la solution retenue	Identifie plusieurs solutions et en crée de nouvelles, fait une analyse critique, basée sur des critères de sélection pertinents et à l'aide d'outils servant à la prise de décision. Valide le potentiel de la solution retenue	Identifie plusieurs solutions et en crée de nouvelles, fait une analyse critique et applique un processus rigoureux et rationnel à l'aide d'outils servant à la prise de décision. Valide le potentiel de la solution retenue et cherche à l'améliorer
Sélectionner les techniques, ressources et outils appropriés pour réaliser une tâche donnée	Q05.1	Sélectionne des techniques, ressources et outils qui ne sont pas appropriés pour réaliser une tâche donnée	Sélectionne des techniques, ressources et outils qui sont minimalement appropriés pour réaliser une tâche donnée	Sélectionne les techniques, ressources et outils appropriés, sans toutefois pouvoir justifier ses choix	Sélectionne les techniques, ressources et outils appropriés en pouvant justifier ses choix (en connaît la portée et les limites)	Sélectionne les techniques, ressources et outils appropriés en pouvant justifier ses choix (en connaît la portée et les limites), de même qu'en pouvant inférer la nature du travail à accomplir ou les données à colliger

8 PRATIQUE PROCÉDURALE 1

But de l'activité

Le but de cette activité est de se familiariser avec certains algorithmes de chiffrement, la théorie mathématique qui explique leur fonctionnement, et les limites de leur utilisation :

- RSA : système à clé publique
- DH : partage sécuritaire de clé privée
- AES : méthode de chiffrement symétrique

NOTE : Il vous est fortement recommandé de lire les documents qui expliquent RSA, la théorie des nombres et le chiffrement symétrique avant de vous présenter à cette séance.

Pour faire les exercices procéduraux qui suivent, vous devrez faire certains calculs qui pourraient s'avérer exigeants si vous les faites seulement avec un papier et un crayon. Il vous est donc recommandé d'utiliser une calculatrice ou une application logicielle de calcul qui supporte les grands nombres.

8.1 EXERCICES

P1.E1 RSA - Calcul de clé et chiffrement

Soit $p = 5$, $q = 11$,

a. Quel est n ?
b. Est-il possible de trouver une paire de clés avec $e = 59$? Si oui, quel est le d correspondant ? Si vous avez trouvé d, calculez $(d^{-1} \mod \phi(n))$ et $(d^{-1} \mod \lambda(n))$. Expliquez vos résultats.
c. Est-il possible de trouver une paire de clés où e est un nombre pair ? Pourquoi ?
d. Est-il possible de trouver une paire de clés avec $e = 5$, 15, 25 ou 35 ? Si oui, quel est le d correspondant à chacun des e ?
e. Est-il possible de trouver une paire de clés avec $e = 9$, 11, 19, 21, 29, 31 ou 39 ? Si oui, quel est le d correspondant à chacun des e ? Est-ce que ce sont vraiment des clés de chiffrement ? Pourquoi ?
f. Pour $e = 1$, quel est d ? Est-ce qu'ils forment une paire de clés valable ? Pourquoi ?
g. Est-il possible d'énumérer la liste de paires de clés valables $(n, e), (n, d)$ pour ce n ? Si oui, quelles sont-elles ? Si non, pourquoi ?
h. L'une des paires de clés possibles est bâtie à partir de $e = 3$. Que devient le nombre 30 lorsqu'on le chiffre avec la clé publique $(n, 3)$? Peut-on déchiffrer le résultat ?

i. Avec la même paire de clés, chiffrez le nombre 85. Peut-on déchiffrer le résultat ? Si oui, comment ? Si non, qu'est-il arrivé ? (Vous pouvez aussi essayer de chiffrer/déchiffrer 140 et 195)
j. Chiffrez ($m = 54$) avec $(n, e) = (55, 3)$. Commentez le résultat.
k. Chiffrez ($m = 11$, 22, 33) avec $(n, e) = (55, 3)$. Qu'ont en commun les résultats ? Pouvez-vous expliquer pourquoi ? (Indice : le TRC peut s'avérer très utile...)
l. Prouvez que $a^{(p-1)} \equiv 1(\mod p)$ (où p est un nombre premier) n'est pas vrai si a est un multiple de p (c'est-à-dire, $a = kp$).

P1.E2 RSA - Utilisation du théorème du reste chinois (problème de Sun Zi)

Nous avons une certaine quantité d'objets, mais nous n'en connaissons pas le nombre exact :

Si nous les comptons par groupes de 5, il nous en reste 3.
Si nous les comptons par groupes de 7, il nous en reste 0.
Si nous les comptons par groupes de 11, il nous en reste 10.
Enfin, si nous les comptons par groupes de 13, il nous en reste 8.

Combien y a-t-il d'objets ? Comment le tuteur devra-t-il modifier ce problème l'an prochain ? Pourrait-on obtenir le même résultat en n'utilisant que les résidus modulo 5, 7 et 11 ?

Question bonus (demande un peu de réflexion) : l'un des étudiants du cours affirme : "Si cette question est encore là quand elle sera en S3 informatique, ma jeune soeur aura comme paramètres du problème des restes de 5 modulo 7 et de 7 modulo 11". En supposant que les étudiants ont habituellement 20 ans en S3, croyez-vous l'affirmation de cet étudiant ? Si oui, en quelle année est née la soeur de l'étudiant ?

P1.E3 RSA - Découverte de clés privées et déchiffrement

a. Soit la clé publique suivante : $(n, e) = (86429, 5)$. Pouvez-vous trouver la clé privée correspondante ? Petit indice : la fonction $f(x) = x^2 + 5$, en commençant avec $x = 1$, pourrait s'avérer utile pour la méthode ρ de Pollard...
b. Vous interceptez un message chiffré avec la clé précédente. Pouvez-vous le déchiffrer ? Le message chiffré est : 30270 Vous avez une idée : peut-être le message original était-il codé en ASCII, avec des blocs de 8 bits accolés les uns aux autres ? Pour en avoir le coeur net, allez consulter le site suivant : http://www.asciitable.com/

P1.E4 Diffie-Hellman : sécurité de la clé privée partagée

Soit $p = 17$, le nombre premier que nous utiliserons pour faire une échange Diffie-Hellman.

a. Soit $g = 13$. Combien y a-t-il de puissances distinctes de g ? Est-ce que leur nombre est suffisant ? (Utiliser la table des puissances de $g \mod 17$ ci-après).

b. Quels sont les meilleurs choix pour g ? Pourquoi ?

c. Quelles sont les tailles des cycles des puissances des différents g ? Quel est leur lien avec $p - 1$?

g	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	2	4	8	16	15	13	9	1	2	4	8	16	15	13	9	1
3	1	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1
4	1	4	16	13	1	4	16	13	1	4	16	13	1	4	16	13	1
5	1	5	8	6	13	14	2	10	16	12	9	11	4	3	15	7	1
6	1	6	2	12	4	7	8	14	16	11	15	5	13	10	9	3	1
7	1	7	15	3	4	11	9	12	16	10	2	14	13	6	8	5	1
8	1	8	13	2	16	9	4	15	1	8	13	2	16	9	4	15	1
9	1	9	13	15	16	8	4	2	1	9	13	15	16	8	4	2	1
10	1	10	15	14	4	6	9	5	16	7	2	3	13	11	8	12	1
11	1	11	2	5	4	10	8	3	16	6	15	12	13	7	9	14	1
12	1	12	8	11	13	3	2	7	16	5	9	6	4	14	15	10	1
13	1	13	16	4	1	13	16	4	1	13	16	4	1	13	16	4	1
14	1	14	9	7	13	12	15	6	16	3	8	10	4	5	2	11	1
15	1	15	4	9	16	2	13	8	1	15	4	9	16	2	13	8	1
16	1	16	1	16	1	16	1	16	1	16	1	16	1	16	1	16	1

P1.E5 AES - Calcul et utilisation

Pour répondre à cette question, le vidéo suivant (moins de 3 minutes) peut être utile. La description disponible sur wikipédia est aussi intéressante.

a. À quoi sert le "*key schedule*" de Rijndael ?
b. Que représente l'état (*state* en anglais) pendant les calculs AES ?
c. Quelle est la fonction des "tours" de calcul (*rounds* en anglais) ?

P1.E6 RSA - Calcul de p et q à partir de n et phi(n)

On vous donne n et $\phi(n)$. Trouvez p et q. Indice : Travaillez l'équation : $(x-p)(x-q) = 0$, en tentant de remplacer les occurrences de p et q par des fonctions de n et $\phi(n)$. Vous obtiendrez une équation de la forme $x^2 + bx + c = 0$, dont les racines seront p et q. Pour vérifier votre solution, calculez p et q pour $n = 218791$ et $\phi(n) = 217800$.

9 PRATIQUE EN LABORATOIRE

Dans cette activité, on collabore par équipe de deux.

But de l'activité

Le but de cette activité est d'identifier les vulnérabilités d'un système informatique, et d'y apporter des correctifs.

Description du laboratoire :

Au laboratoire, à l'aide de 'wireshark', 'ssh', 'ddd', 'objdump', 'hexedit' (ou 'khexedit'), 'mini_crack' et 'python' :

- Trouver le login et le mot de passe d'un usager qui utilise telnet et/ou ftp ;
- Tenter de trouver le login et le mot de passe d'un usager qui utilise ssh et/ou sftp ;
- Trouver les mots de passes d'usagers présents dans le fichier shadow fourni ;
- Observer et comprendre un problème de dépassement de tampon (buffer overflow)
- Générer efficacement des nombres premiers utiles pour RSA

Attention : sauvegarde des données et des modifications effectuées

Il est important de réaliser que le fichier .iso fourni représente un "*Live-CD*", c'est-à-dire une image d'un système d'exploitation qui n'utilise pas le disque. Toute l'information modifiée se trouve en mémoire : le système utilise un système de fichier à union (*union file system*) dont la couche supérieure, qui est modifiable, se trouve en mémoire. Ceci implique que l'arrêt de la machine virtuelle efface toutes les modifications effectuées pendant l'exécution de la machine. Pour ne pas perdre les fichiers que vous avez créés ou modifiés, vous pouvez soit mettre la machine virtuelle en pause (et son redémarrage ultérieur retrouvera vos données), ou bien vous devez sauvegarder vos données sur une machine externe. Si vous désirez sauvegarder vos données, voici les étapes à suivre :

1. Dans un terminal, se déplacer vers le répertoire de knoppix :

» **cd**

2. Créer une archive avec tout ce que vous avez modifié (dans tmp) :

» **tar cvfz sauvegarde.tar.gz ./tmp**

3. Utiliser sftp pour vous connecter à votre compte sur ssh.gel.usherbrooke.ca :

» **sftp votre_cip@ssh.gel.usherbrooke.ca**

4. Copier l'archive :

» **put sauvegarde.tar.gz**

Pour retrouver vos données (après un arrêt et un redémarrage de la machine virtuelle Sherbrix), vous devez effectuer les même étapes, mais à rebours :

1. Dans un terminal, se déplacer vers le répertoire de knoppix :
» **cd**
2. Utiliser sftp pour vous connecter à votre compte sur ssh.gel.usherbrooke.ca :
» **sftp votre_cip@ssh.gel.usherbrooke.ca**
3. Reprendre l'archive :
» **get sauvegarde.tar.gz**
4. Extraire le contenu de l'archive :
» **tar xvfz sauvegarde.tar.gz .**

9.1 Détection de login et de mot de passe sur le réseau

Dans cet exercice, vous utiliserez *wireshark* pour tenter d'identifier des noms d'usagers et leurs mots de passe. Vous devez utiliser le fichier ".iso" Knoppix fourni (sherbrix). Knoppix permet à un usager de démarrer GNU/Linux sur un PC sans avoir à installer le système d'exploitation sur le disque dur de la machine. Lorsque le système est démarré, vous pouvez démarrer *wireshark* à partir du menu KDE (accessible en bas à gauche), sous la rubrique *internet*. Pour la capture, vous devriez écouter le *loopback*, qui apparaît sous le nom *lo* dans le champ *interface* lorsque vous démarrez la capture de paquets (*Menu Capture->Start*).

Lorsque *wireshark* est démarré, et que la capture est en route, vous devriez observer que différentes connexions prennent place. C'est le consultant externe de chez Mégamillions, qui se connecte régulièrement au serveur de l'entreprise (à vue de nez, il se connecte environ 1 fois par minute...). Quels types de connections observez-vous ? Pouvez-vous identifier le nom de l'usager et son mot de passe au travers des paquets captés ? La commande "Follow TCP Stream" de Wireshark pourrait vous être utile...

Tentez maintenant d'utiliser *ssh* pendant la capture de paquets par *wireshark*. La commande *ssh localhost* devrait faire l'affaire. De même, utilisez *sftp* pendant la capture de paquets. Consultez le fichier */etc/services*, pour découvrir la liste de ports possiblement ouverts sur votre machine, et voyez quelles applications y sont associées.

9.2 Identification de mots de passe

Sur le système knoppix qui vous a été fourni, vous trouverez le fichier *mini_crack.c* sous */home/knoppix/tmp/shadow_crack*. Dans ce répertoire, vous trouverez différents fichiers,

dont *shadow*, *caracteres*, et *mots_de_passe*. Le fichier *shadow* est une version abrégée du fichier */etc/shadow* se trouvant sur votre système (utiliser *man shadow* pour en connaître plus au sujet de ce fichier). En ajoutant le code nécessaire au fichier *mini_crack.c*, vous serez capables de percer les mots de passes des trois usagers inclus dans ce fichier shadow, à l'aide du dictionnaire et du fichier de caractères fournis, en faisant une recherche exhaustive de mots de passe contenant 3 caractères. Consultez le commentaire au début du fichier *mini_crack.c* pour avoir des indices sur le code que vous devez ajouter pour faire fonctionner cette application (*man crypt* pourrait s'avérer utile...).

Lorsque vous aurez ajouté le code manquant, vous pourrez compiler le tout à l'aide de *make_mini_crack.sh*, puis utiliser les scripts *run_mini_crack.sh* et *run_mini_crack_verbose.sh* pour avoir un exemple de l'utilisation de ce petit programme. Modifiez les fichiers de paramètres pour faire vos propres recherches de mots de passe sur le fichier *shadow* qui se trouve dans le répertoire.

a. Créez un fichier de caractères pour la recherche exhaustive qui contienne tous les chiffres, toutes les lettres (minuscules et majuscules), ainsi que les signes de ponctuation et les autres caractères utilisables dans un mot de passe. Combien de caractères avez-vous ? (Vous pouvez utiliser *wc* pour établir rapidement ce nombre).
b. Créez une copie du fichier *shadow*, avec une seule entrée, Puis faites une recherche exhaustive avec 2 caractères, sans utiliser de dictionnaire. Combien de temps prend votre recherche ? Pour faire une bonne mesure du temps d'exécution, vous pouvez utiliser la commande *time* de la façon suivante : *time mini_crack (options)*. Vous aurez ainsi le temps d'exécution à la fin.
c. Essayez maintenant avec 3 caractères. Quel est le temps d'exécution ?
d. Pouvez-vous évaluer le temps requis pour une recherche à 4, 5, 6, 7, 8, 9, 10 caractères ?
e. En tenant compte des calculs faits en d, évaluez pour combien de caractères (en fonction du nombre d'entrées dans le fichier *shadow*) il est possible de faire une recherche exhaustive qui se terminera en dedans de quelques minutes.
f. En vous basant sur les calculs faits en d, évaluez la taille maximale du dictionnaire que vous pouvez utiliser, s'il n'y a pas de recherche exhaustive. Ce résultat devrait être en fonction du nombre d'entrées dans le fichier *shadow*.

9.3 Problème de dépassement de tampon (buffer overflow)

Observez le code du programme *test_buf.c*, situé dans le répertoire */home/knoppix/tmp/buffer_overflow*, et effectuez les commandes suivantes de l'intérieur de ce répertoire :

./test_buf good_input
./test_buf bad_input

a. Est-ce que les deux exécutions se comportent différemment ?
b. Lisez le code source, et tentez de voir comment le message *" Ce message ne devrait jamais etre affiche... "* peut se faire imprimer.
c. Faites une copie *bad_input2* du fichier *bad_input*, et effacez les 20 derniers caractères. Effectuez la commande suivante, et observez ce qui se passe : *./test_buffer bad_input2*
d. Ajoutez 20 caractères arbitraires à votre fichier *bad_input2*, et effectuez de nouveau la commande précédente. Que se passe-t-il ? Pouvez-vous l'expliquer ?
e. Utilisez *wc* (word count) pour déterminer la taille du fichier *bad_input*. Pouvez-vous trouver un lien entre cette taille et certains éléments dans le fichier source *test_buf.c* ?
f. Utilisez la commande *ddd test_buf* pour pouvoir exécuter le programme pas à pas. Dans *ddd*, utilisez les commandes suivantes :

   b main (*break main* - arrête l'exécution dans la procédure main)

   b appel_problematique (arrête dans cette procédure)

   n (*next* - exécute une nouvelle ligne dans le programme, saute les appels de procédure)

   s (*step* - comme n, mais entre dans les appels de procédure)

   c (*continue* - continue l'exécution)

   p nom_de_variable (imprime la valeur de la variable)

   info frame (donne l'information sur le cadre d'exécution d'une procédure. Faites attention au *" saved eip "*, qui contient l'adresse de retour)

   r bad_input (démarre l'exécution avec le paramètre *bad_input*. Par la suite, on peut tout simplement utiliser *r*, puisque *ddd* réutilise les paramètres de démarrage de l'exécution précédente)

   Examinez ce qui se passe lorsque vous faites un *step* sur le return de la procédure *appel_problematique*. Où va l'exécution ?

g. Utilisez la commande suivante : *objdump −−source test_buf | less*. Vous verrez le code assembleur du programme, entremêlé avec les lignes de code du programme source. Observez en particulier les adresses apparaissant devant chacune des lignes assembleur. Dans une autre fenêtre de commande, utilisez maintenant la commande suivante : *hexedit bad_input* (ou *khexedit bad_input*, qui donne une interface graphique)(pour lire et modifier le fichier en format hexadécimal). Allez voir vers la fin du fichier, et observez les codes hexadécimaux des caractères qui apparaissent un peu avant la fin du fichier. Pouvez vous trouver un lien entre ces caractères et les adresses qu'on voit dans *objdump* ? Notez que les nombres ne sont pas nécessairement représentés dans l'ordre des octets *bytes* qu'on attendrait. Le "*endianness*" des nombres peut s'avérer important (les lectures à ce sujet pourraient s'avérer intéressantes).

h. Faites une copie *bad_input3* de *bad_input*. Pouvez-vous la modifier de sorte que le message *" Pouvez-vous faire afficher ce message ? "* apparaisse à l'écran ?

9.4 Chiffrement par flux

Observez le code du programme *flux.py*, situé dans le répertoire */home/knoppix/tmp/flux*. Ce script Python est destiné à faire le chiffrement par flux d'un fichier externe, à l'aide d'une clé symétrique. Complétez les fonctionnalités manquantes pour réussir à déchiffrer le fichier *test.txt* présent dans le même répertoire. Le fichier est traité un caractère à la fois, c'est-à-dire par blocs de 8 bits. Un caractère du fichier à chiffrer est combiné avec un caractère de la clé, à l'aide d'un opérateur OU-Exclusif (bitwise xor)) : le plus petit octet de la clé est utilisé, pour se combiner avec le caractère courant. Puis, on décale la clé de 8 bits et on recommence avec le caractère suivant. S'il reste des caractères à traiter mais que la clé est rendue nulle, on recommence avec la clé initiale (voir figure 9.1).

La taille du fichier à déchiffrer est plus grande que la taille de la clé, qui a été répétée lors du chiffrement. Croyez-vous que c'était une bonne décision pour ce qui est de la sécurité du chiffrement ?

Note : Vous pouvez ouvrir une fenêtre avec l'interpréteur Python interactif en utilisant *Python (v2.4)* dans le menu *Development*. Vous pouvez aussi appeler directement l'interpréteur Python dans une fenêtre interactive.

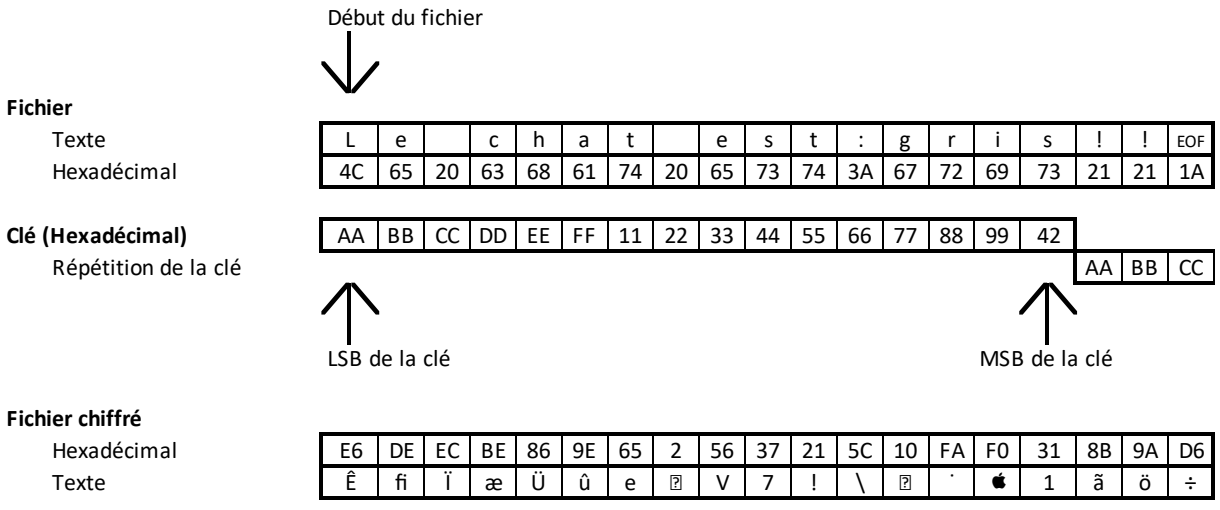

FIGURE 9.1 Méthode de chiffrement par flux à coder

9.5 RSA - Calcul de nombres premiers très grands

Soit $PPNP35(x)$ le plus petit nombre premier plus grand ou égal à x, qui peut être utilisé pour produire une clé RSA où e peut être égal à 3 ou 5.

a. Modifiez l'algorithme de génération de nombres premiers pour produire $PPNP35(x)$ et expliquez son fonctionnement.
b. Le nombre produit par votre algorithme est-il un nombre premier avec une certitude absolue ? Pourquoi ?
c. Trouvez $PPNP35(10^{30})$
d. Trouvez $(PPNP35(10^{200}) - 10^{200})$

9.6 Problème facultatif : dépassement de tampon (buffer overflow)

Extrayez le contenu du fichier */home/knoppix/tmp/divers/canari/canari.tar*. En utilisant le fichier */home/knoppix/tmp/buffer_overflow/good_input*, vérifiez que *test_buf_canari* se comporte comme *test_buf* avec un fichier de taille acceptable. Ensuite, utilisez la commande *'test_buf_canari bad_input_canari'* pour tenter d'utiliser le problème de buffer overflow. Vous pouvez vérifier que l'adresse de retour de *'appel_problematique'* est effectivement modifiée, et que la nouvelle adresse de retour correspond à un appel à la procédure *'system'*. (Vous pouvez vérifier ce qui se passe lorsqu'un bon fichier est donné en paramètre). Qu'arrive-t-il lors de l'utilisation du mauvais fichier ? Pourquoi est-ce que l'appel à la procédure *'system'* ne se fait-elle pas ? En observant de près l'exécution du code (avec *ddd*), pouvez-vous modifier le fichier *bad_input_canari* pour pouvoir utiliser le problème de buffer overflow ? Si oui, comment, et si non, pourquoi ?

10 PRATIQUE PROCÉDURALE 2

But de l'activité

Étudier différentes solutions aux problèmes de sécurité étudiés (canaris, stack non-exécutable, tunnels VPN), comprendre comment on utilise la méthode de chiffrement RSA.

10.1 Virtual Private Networks (VPNS)

Vous avez fort probablement déjà utilisé des réseaux virtuels privés (RVP en français, VPN en anglais), par exemple pour vous connecter aux serveurs internes de l'université à partir de la maison. Les questions qui suivent se rapportent à ce type de solution aux problèmes de la visibilité des paquets sur le réseau internet.

a. Sur le réseau internet, les communications chiffrées font souvent usage de deux types de chiffrement, l'un de type RSA, et l'autre de type AES. Par exemple, Alice va produire une clé AES, la chiffrer avec la clé publique RSA de Bob, et lui envoyer cette clé chiffrée. De son côté, Bob fera de même. Pourquoi procède-t-on de cette façon ?
b. Qu'arrive-t-il si Ève envoie à Alice une clé publique RSA en prétendant que cette clé provient de Bob ?
c. Comment Ève pourrait-elle écouter la communication entre Alice et Bob sans que ceux-ci ne le sachent ?
d. Que peut-on faire pour empêcher Ève de procéder ainsi ?

10.2 RSA - Considérations théoriques

Lorsqu'on fait des calculs de modulo avec des nombres ayant de très gros exposants, on décompose l'exposant en multiples de 2, et on utilise des modulos de nombres intermédiaires (par exemple, $V^4 \mod (n) = ((V^2 \mod (n)) \times (V^2 \mod (n))) \mod (n)$).

a. Soit : $V = kn + r$, avec V, k, n et r entiers, et $r < V$; Pouvez-vous prouver que $V^2 modulo(n) = r^2 modulo(n)$?
b. Soit : $T = UW$, avec T, U, W entiers. Prouvez que $T \mod (n) = ((U \mod (n)) \times (W \mod (n))) \mod (n)$

10.3 Chiffrement Diffie-Hellman

Dans cette question, nous étudions l'impact du choix de p et g sur la sécurité de la méthode Diffie-Hellman. Pour aider votre réflexion dans cette question, vous pouvez observer le tableau des puissances de $(g \mod 17)$ utilisé lors du premier procédural.

a. Soit q un très grand nombre premier, et $p = 2 \times p_1 \times p_2 \cdots \times p_k \times q + 1$ un autre nombre premier, avec $p_1, p_2, \ldots, p_k$ également des nombres premiers. Quelles sont les tailles des cycles de puissances possibles pour $g^x \mod p$, avec $1 \leq g < p$ et $x = 1, 2, 3, \ldots$ un nombre entier ? Comment pourrait-on valider que les puissances successives d'un certain g forment un cycle de taille t ?

b. Soit $p = 2 \times q + 1$, où p et q sont des nombres premiers. Quelles sont les longueurs possibles des cycles des puissances de g (où $1 \leq g < p$) ? Le petit théorème de Fermat peut être utile pour répondre à cette question.

   Note : les nombres premiers q pour lesquels $p = 2 \times q + 1$ est aussi un nombre premier sont appelés des nombres premiers de Sophie Germain en l'honneur de la mathématicienne française qui les a étudié.

10.4 Chiffrement Diffie-Hellman 2

Vous recevez plusieurs triplets Diffie-Hellman d'un interlocuteur. Évaluez la validité de ces triplets, et expliquez vos réponses.

a. $(q, p, g) = (1237, 19801, 13)$
b. $(q, p, g) = (1007, 6043, 9)$
c. $(q, p, g) = (12347, 123471, 11)$
d. $(q, p, g) = (1009, 10091, 3)$

10.5 Factorisation - Méthode (p-1) de Pollard

Tentez de factoriser le nombre 482723 à l'aide de la méthode $(p - 1)$ de Pollard. Pour ce faire, vous pouvez explorer les puissances successives du nombre 2 (beaucoup d'autres nombres peuvent être utilisés, mais le calcul sera plus facile avec 2, puisque pour les premières élévations à une puissance, vous conserverez un nombre qui est un multiple de 2. Ceci devrait vous permettre de sauter quelques étapes de calcul (lesquelles, et pourquoi ?)

11 VALIDATION AU LABORATOIRE

Le but de cette activité est de vous permettre de valider la solution proposée à la problématique de cette unité.

- Assurez-vous d'avoir bien compris les différentes étapes requises pour résoudre la problématique.
- Vous serez questionnés au sujet de ces étapes, des mathématiques sous-jacentes et des techniques utilisées.
- Vous devrez réserver une période de 10 minutes à partir d'un lien fourni dans les jours précédents sur le site de l'unité d'APP.