M2-TIW4 sécurité des systèmes d'informations
Contrôle continu final – durée 1h30

Master Technologie de l'Information, promotion 2015 – 2016

Mercredi 16 décembre 2015

Aucun document autorisé. Le barème est indicatif. Ne pas rendre le sujet. La concision, la précision et la clarté des réponses font partie intégrante de l'évaluation. Ne pas rendre le sujet avec la copie.

Exercice 1 : Pot-pourri (/4)

Pour les questions 2, 3 et 4 indiquer la ou les réponses correctes. -0.5 par erreur.

1. On donne les trois catégories principales de la classification SANS des erreurs logicielles : (A) *Insecure Interaction Between Components*, (B) *Risky Resource Management* et (C) *Porous Defenses*. Pour chacune des erreurs suivantes, indiquer la catégorie à laquelle elle se rattache.

   ☐ *Use of a One-Way Hash without a Salt* ;

   ☐ *Incorrect Calculation of Buffer Size* ;

   ☐ *URL Redirection to Untrusted Site* ;

   ☐ *Use of a Broken or Risky Cryptographic Algorithm* ;

2. Un moniteur de contrôle d'accès doit *toujours* disposer de la propriété...

   ☐ Être vérifiable (on peut prouver qu'il prend bien les bonnes décisions) ;

   ☐ Être sans mémoire (la décision ne dépend pas de l'historique des accès précédents) ;

   ☐ Être temp-réel (la décision doit être prise dans un temps limité) ;

   ☐ Être inévitable (on ne peut pas contourner le moniteur) ;

3. Le contrôle d'accès...

   ☐ Discrétionnaire ne suppose pas d'autorité qui régit les droits de tous ;

   ☐ Mandataire ne suppose pas d'autorité qui régit les droits de tous ;

   ☐ Discrétionnaire est traditionnellement utilisé dans les systèmes de fichiers Linux ;

   ☐ Mandataire est traditionnellement utilisé dans les systèmes de fichiers Linux.

4. Avec la hiérarchie *Très Secret Défense* (TS), *Secret Défense* (S), *Confidentiel Défense* (C) et *Non-classifié* (U), on suppose que l'on a deux utilisateurs u_0 et u_1 accrédités respectivement C et S ainsi que deux fichiers f_0 et f_1 auxquels sont associés les niveaux respectifs TS et C.

   ☐ u_0 peut lire et écrire dans f_1 ;

   ☐ u_1 peut écrire dans f_0 ;

   ☐ u_1 peut écrire dans f_1 ;

   ☐ si on accrédite u_1 à C, les droits restent les mêmes.

Pour les deux exercices suivants, on s'intéresse à un dossier médical personnel partagé reparti et sécurisé, stocké sur un dispositif portable appelé *token*. Ce *token* est une clef USB de 16Go à laquelle est intégrée un petit processeur et un coprocesseur cryptographique, capable d'effectuer du (dé)chiffrement AES de façon efficace. En plus de la mémoire RAM usuelle, le *token* dispose d'une petite zone de mémoire sécurisée, dans laquelle sont stockées les clefs utilisées par le coprocesseur. Les 16Go de mémoire de masse ne sont pas sécurisés (par exemple, c'est une carte SD). Il faut donc chiffrer les données que l'on y écrit pour assurer la confidentialité en cas de perte du *token*.

Un *token* peut stocker des données de son propriétaire mais aussi des données d'autres personnes. C'est typiquement le cas des *token* des professionnels de santé (pour les dossier des patients qu'ils traitent), mais aussi pour les patients eux-mêmes (on pourrait avoir un *token* pour toute une famille, ou stocker une partie du dossier d'un proche).

Dans ce système de dossier médical, chaque participant A dispose d'une paire RSA (K_A, K_A^{-1}) où K_A est la partie privée et K_A^{-1} la partie publique. L'autorité centrale C (disons, la sécurité sociale) distribue à chaque participant A un certificat C_A qui est stocké dans son *token*. Les participants sont de deux types différents : soit de type *patient* (PA), soit de type *professionnel de santé* (PS). C'est C qui définit le type de chaque participant. Le *token* est utilisé par les professionnels et les patients pour s'authentifier les uns les autres, en plus de la fonctionnalité de stockage du dossier médical. Chaque *token* stocke, en mémoire sécurisée, une clef AES 256 bits notée M_A qui est appelée *clef maître*. La clef M_A permet de chiffrer K_A. Cette clef maître permet aussi de chiffrer le dossier médical stocké dans la mémoire de masse.

Exercice 2 : Sauvegarde de la clef maître d'un *token* (/8)

On souhaite définir un protocole de sauvegarde des données du *token* en utilisant le schéma de partage de clé secrète de Shamir. Dans ce protocole, un patient A va envoyer l'intégralité de son dossier chiffré sur un serveur dans le cloud. Pour la sauvegarde confidentielle de M_A, qui permettra de recouvrir un dossier stocké dans le cloud, on souhaite faire en sorte qu'il faut le concours d'au moins un professionnel de santé et de deux personnes choisies par A pour arriver à retrouver intégralement la clef M_A. Le texte suivant est extrait de Wikipedia [1].

> *Le partage de clé secrète de Shamir (*Shamir's Secret Sharing*) est un algorithme de cryptographie. C'est une forme de partage de secret, où un secret est divisé en parties, donnant à chaque participant sa propre clé partagée, où certaines des parties ou l'ensemble d'entre elles sont nécessaires afin de reconstruire le secret.*
>
> *Formellement, notre objectif est de diviser certaines données D (par exemple, la combinaison du coffre) en n pièces $D_1, \dots, D_n$ de telle sorte que :*
> *— la connaissance de k ou plus D_i pièces rend D facilement calculable.*
> *— la connaissance de $k-1$ ou moins D_i pièces rend D complètement indéterminée.*
> *Ce régime est appelé schéma de seuil $(k; n)$. Si $k = n$ alors tous les participants sont nécessaires pour reconstituer le secret. L'idée essentielle d'Adi Shamir est que 2 points sont suffisants pour définir une ligne, 3 points suffisent à définir une parabole, 4 points pour définir une courbe cubique, etc. Autrement dit, il faut k points pour définir un polynôme de degré $k-1$.*
>
> *Supposons que nous voulons utiliser un schéma de seuil $(k; n)$ pour partager notre secret S, que l'on suppose, sans perte de généralité, être un élément dans un corps fini F de cardinalité P avec P un nombre premier, $0 < k \leq n < P$ et $S < P$. Les étapes du protocole sont les suivantes :*
>
> 1. *Choisir au hasard $(k-1)$ coefficients $a_1, \dots, a_{k-1}$ dans F, et poser $a_0 = S$.*
> 2. *Construire le polynôme $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_{k-1}x^{k-1}$.*
> 3. *Choisir au hasard n valeurs de F, par exemple $v_1, \dots, v_n$ et distribuer un couple $(v_i, f(v_i))$ à chacun des n participants.*

1. https://fr.wikipedia.org/wiki/Partage_de_cl%C3%A9_secr%C3%A8te_de_Shamir

Étant donné un sous-ensemble de k de ces couples, nous pouvons trouver les coefficients du polynôme f à l'aide de l'interpolation polynomiale, le secret étant le terme constant $a_0 = f(0)$.

1. On considère le cas du schéma de seuil $(2, n)$ avec $P = 127$. On connaît les couples $(1, 106)$ et $(4, 44)$. Retrouver le secret S[2]. (/1)
2. Pourquoi tirer une clef maître plutôt que chiffrer directement le dossier médical avec la clef RSA ? (/1)
3. Définir ce que contient C_A en utilisant la notation usuelle[3]. (/1)
4. Le protocole de sauvegarde que l'on souhaite concevoir garantit-il la confidentialité, l'intégralité ou la disponibilité du dossier médical ? Justifier votre réponse pour chaque critère en une phrase maximum. (/1)
5. Lorsqu'un professionnel souhaite écrire des données sur un dossier patient, les *tokens* doivent d'abord vérifier qu'ils communiquent bien entre les bonnes personnes grâce aux certificats. Donner les étapes d'un protocole d'authentification mutuelle à clef publique en utilisant la notation usuelle. Penser à vérifier le type des participants. (/1)
6. On remarque que l'on peut combiner plusieurs itérations du système de Shamir pour faire en sortie que ce soit *au moins un professionnel de santé* et *deux personnes choisies ensembles* qui soient nécessaires à la récupération de M_A. Expliquer comment faire. (/2)
7. Un utilisateur perd son *token*. Expliquer quelles sont les étapes à suivre pour qu'il retrouve l'intégralité de son dossier. (/1)
8. (Bonus /1) Avec $P = 127$, quelle est la taille maximum des clefs que l'on peut espérer partager ? On désire sauvegarder M_A. Quelle valeur de P doit-on choisir ?
9. (Bonus /1) En fait ce n'est pas M_A qui est utilisé pour chiffrer le dossier médical, mais une clef dérivée de M_A. Donner une méthode pour dériver n clefs différentes à partir de M_A.

Le *token* permet de développer des applications *réputées sûres* : le code doit être signé par C et le *token* refusera d'exécuter du code non signé. Le *token* est notamment capable d'exécuter un Système de Gestion de Base de Données (SGBD, comme par exemple *SQLite*) qui stocke ses tables dans la mémoire de masse. Ce moteur sert entre autres à indexer les fichiers et à gérer les droits des personnes qui y accèdent.

Exercice 3 : Gestion des droits au dossier de santé (/8)

Le dossier médical est une collection de fichiers (des documents, des images médicales). L'index des fichiers est une relation `Index(IDFile, IDPatient, IDCreator)`, où `IDFile` est clef et où les attributs ont la sémantique intuitive attendue. L'attribut `IDPatient` désigne l'identifiant de l'unique patient concerné par le fichier. Le SGBD a aussi une relation `Group(IDFile, IDGroup)` qui permet de gérer l'organisation en groupes sémantiques des fichiers. Elle remplace l'arborescence usuelle d'une système de gestion de fichier. On note qu'un fichier peut appartenir à plusieurs groupes. Des exemples de groupes sont *diabète chronique*, *grippe 2014* ou encore *gestation 2015*.

On souhaite développer un modèle de contrôle d'accès sur mesure pour l'accès aux fichiers qui constituent un dossier médical. Les droits d'accès seront stockés et contrôlés par le moteur de base de données qui joue ici le rôle de moniteur de contrôle d'accès aux fichiers de la mémoire de masse. Ici, une requête d'accès reçue par un *token* est un quadruple `(IDSubject, Status, IDFile, action)` où `IDSubject` est l'identifiant de l'accédant et `action` est soit `read` soit `write`. L'attribut

2. Pour diviser a par b modulo P, on va chercher le plus petit entier congru à a modulo P divisible par b. Par exemple pour $a = 65$ et $b = 3$ on obtient $(65 \div 3) \mod 127 = ((65 + 127) \div 3) \mod 127 = (192 \div 3) \mod 127 = 64 \mod 127 = 64$. Pour la soustraction, c'est similaire : $3 - 65 \mod 127 = (127 + 3) - 65 \mod 127 = 130 - 65 \mod 127 = 65$.

3. Pour rappel, on note $\{\langle x, h(y)\rangle\}_{K_A^{-1}}$ la concaténation d'un message x avec le haché d'un message y, le tout chiffré avec la clef publique de A.

`Status` est soit `PA` pour un patient, soit `PS` pour un professionnel, soit `Alert` pour un professionnel intervenant en état d'urgence.

Le modèle de contrôle d'accès doit permettre d'assurer la politique de sécurité suivante :

1. le créateur d'un fichier doit toujours pouvoir lire et modifier un fichier qu'il a crée ;
2. tous les professionnels peuvent toujours créer des fichiers ;
3. un patient peut toujours créer des fichiers dans son token ;
4. le patient concerné par un fichier peut toujours le lire ;
5. par défaut, sauf pour les cas précédents, personne ne peut lire un fichier ;
6. en cas d'urgence, un professionnel peut accéder à tous les fichiers en lecture, mais alors toutes les opérations qu'il exécute sont tracées par le SGBD.

Quand un fichier est crée, c'est-à-dire pour une requête dont l'action est `write` sur un fichier qui n'existe pas dans l'index, le patient concerné est le propriétaire du *token* où ce fichier est écrit. L'identifiant du propriétaire du *token* est considérée comme une variable globale *owner*. On suppose qu'en parallèle de la requête, le certificat de l'accédant est également transmis, ce qui permet d'authentifier le sujet et vérifier son statut.

Le *logging* des actions en état d'urgence est un *effet de bord* de l'évaluation des droits (comme un affichage console exécuté par une fonction), qui ne change pas la sémantique des droits d'accès. On ne considère pas le contrôle d'accès aux tables du SGBD mais bien l'accès aux fichiers qu'il indexe.

1. Ce système est-il mandataire, discrétionnaire ou hybride ? Justifier votre réponse. (/1)
2. Expliquer pourquoi un patient ne pourra pas profiter de l'état d'urgence pour pouvoir accéder à un dossier médical qui ne lui appartient pas. (/1)
3. Étant donnée une instance $\Sigma = (\mathit{Index}, \mathit{Group})$ de la BD et une requête $Q = (S, L, F, A)$, on souhaite donner un algorithme en pseudo-code[4] de la fonction $\mathit{Eval}(Q, \Sigma)$ qui décide si la requête Q est autorisée (valeur de retour 1) ou refusée (valeur de retour 0) selon l'instance Σ. Pour chacune chacune des règles de la politique de sécurité, formaliser la condition qui décrit quand l'accès est autorisé. Par exemple, pour la première on pourrait écrire : (/3, bonus pour la qualité du pseudo code /1)

$$\mathit{Eval}((S, L, F, A), \Sigma) = 1 \text{ si } S \in \pi_{\mathit{IDCreator}}(\sigma_{\mathit{IDFile}=F}(\mathit{Index}))$$

4. Donner le pseudo-code de la fonction complète $\mathit{Eval}(Q, \Sigma)$. (/1)
5. On souhaiterait ajouter la règle suivante à la politique de sécurité « *un patient concerné par un fichier peut, pour chaque groupe de fichier* et *pour chaque fichier pris individuellement, spécifier la liste des professionnels et des autres patients qui peuvent y accéder en lecture* ». Indiquer quelles tables ajouter au schéma de BD et expliquer (sans le faire) comment modifier $\mathit{Eval}(Q, \Sigma)$. (/2)

4. Utilisant par exemple les notations mathématiques usuelles, de la programmation impérative, fonctionnelle ou logique et l'algèbre relationnelle.

Corrections

Solution de l'exercice 1

1. Respectivement C, B, A, C.
2. Respectivement vrai, faux, faux, vrai.
3. Respectivement vrai, faux, vrai, faux.
4. Respectivement vrai, vrai, faux, faux.

Solution de l'exercice 2

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 $M_A^n = h^n(M_A)$.

Solution de l'exercice 3

1. Hybride : la détermination du type PS ou PA est mandataire car entièrement sous le contrôle de autorité C. En revanche, la partie des droits spécifiques est à la discrétion du patient propriétaire.
2. Car seuls les professionnels peuvent l'utiliser, et à l'authentification on vérifiera que l'accédant est bien PS.
3. Soit $Q = (S, L, F, A)$ une requête d'accès. On formalise d'abord la politique :

1. le créateur d'un fichier doit toujours pouvoir lire et modifier un fichier qu'il a crée ;
   $Eval((S, L, F, A), \Sigma) = 1$ si $S \in \pi_{IDCreator}(\sigma_{IDFile=F}(Index))$
2. tous les professionnels peuvent toujours créer des fichiers ;
   $Eval((S, L, F, A), \Sigma) = 1$ si $L = PS \wedge A = r \wedge F \notin \pi_{IDFile}(\sigma_{IDFile=F}(Index))$
3. un patient peut toujours créer des fichiers dans son token ;
   $Eval((S, L, F, A), \Sigma) = 1$ si $L = PA \wedge A = w \wedge S = owner \wedge F \notin \pi_{IDFile}(\sigma_{IDFile=F}(Index))$
4. le patient concerné par un fichier peut toujours le lire ;
   $Eval((S, L, F, A), \Sigma) = 1$ si $L = PA \wedge A = r \wedge S \in \pi_{IDPatient}(\sigma_{IDFile=F}(Index))$
5. par défaut, sauf pour les cas précédents, personne ne peut lire un fichier. Ici, pas vraiment de condition, ça sera la valeur par défaut de $Eval$.
6. en cas d'urgence, un professionnel peut accéder à tous les fichiers en lecture ;
   $Eval((S, L, F, A), \Sigma) = 1$ si $L = Alert \wedge A = r$ (et que le certificat de C_S prouve que S est bien un professionnel).

4. En nommant de C_1 à C_6 les conditions précédentes, $Eval((S, L, F, A), \Sigma) = C_1 \vee C_2 \vee C_3 \vee C_4 \vee C_6$
5. On ajoute `AuthGroup(IDGroup, IDUser)` et `AuthFile(IDFile, IDUser)`. Ensuite on ajoute une nouvelle règle qui stipule qu'un accédant S peut lire le fichier F si F est dans un groupe autorisé à S ou directement autorisé à S et on ajoute cette condition C_7 à disjonction de la définition de $Eval$. La condition est formalisée ainsi $Eval((S, L, F, r), \Sigma) = 1$ si $S \in \pi_{IDUser}(\sigma_{IDFile=F}(Index \bowtie AuthFile)) \cup \pi_{IDUser}(\sigma_{IDFile=F}(Index \bowtie Group \bowtie IDGroup))$