Université de Sherbrooke - Département d'informatique
IFT814 – Cryptographie
Martin Fiset, chargé de cours

Devoir 4
Date de remise : Vendredi 14 novembre à 23h59

Effectuez ce devoir en équipe d'au plus 4 personnes. Utilisez TurninWeb, le système de soumission de travaux du Département d'informatique, pour soumettre votre travail. Soumettez un seul fichier pdf par équipe (n'oubliez pas d'y mettre tous les noms), contenant le pseudo-code source nécessaire comme texte dans le corps de votre devoir. Ce devoir compte pour 12% de la note finale.

Contexte

Nous voulons qu'Alice et Bob puissent communiquer de façon sécurisée, disons Alice veut envoyer un message m à Bob de sorte que la transmission de m soit faite de manière confidentielle, intègre et authentifiée.

Ils communiquent sur un réseau non-sécurisé, et donc, en particulier, il est possible qu'un adversaire puisse :

- Lire toutes les communications transmises,
- Modifier les communications,
- Tenter de se faire passer pour Alice,
- Enregistrer des communications passées pour les rejouer (attaque par rejeu),
- Compromettre des clés de session anciennes,
- Forcer l'utilisation d'algorithmes cryptographiques plus faibles (attaque par dégradation)

Ils veulent utiliser les systèmes cryptographiques suivants :

- $\Pi_{CPriv} = (Gen_{CPriv}, E_{CPriv}, D_{CPriv})$, un système de chiffrement à clés privées sécuritaire,
- $\Pi_{MAC} = (Gen_{MAC}, MAC, Verif_{MAC})$, un système d'authentification à clés privées sécuritaire,
- $\Pi_{CPub} = (Gen_{CPub}, E_{CPub}, D_{CPub})$, un système de chiffrement à clés publiques sécuritaire,
- $\Pi_{Sign} = \left(Gen_{Sign}, Sign, Verif_{Sign}\right)$, un système de signatures digitales sécuritaire,
- $\Pi_{KE} = \left(\Pi_A^i, \Pi_B^i\right)$, un système d'échange de clés interactif sécuritaire.

Alice et Bob ont tous deux confiances en Charlie pour émettre des certificats de confiance, et Alice et Bob connaissent tous deux la clé public pk_C de Charlie. Alice a reçu un certificat $cert_{C \to A}$ de Charlie et Bob a reçu un certificat $cert_{C \to B}$ de Charlie.

1. Protocole ''handshake'' avec protection avancée

En utilisant les systèmes cryptographiques $\boldsymbol{\Pi_{CPriv}, \Pi_{MAC}, \Pi_{CPub}, \Pi_{Sign}, \Pi_{KE}}$ ainsi que les certificats $\boldsymbol{cert_{C\to A}}$ et $\boldsymbol{cert_{C\to B}}$ et la clé publique de Charlie $\boldsymbol{pk_C}$, donner le pseudo-code d'un protocole qui permet à Alice et Bob d'établir une clé privée partagée k avec les propriétés suivantes:

- **Robuste :** si les messages transmis de Alice vers Bob et de Bob vers Alice ne sont pas modifiés lors de leur transmission, Alice et Bob n'abandonnent pas le protocole et produisent tous les deux la même clé $\boldsymbol{k}$ en sortie de protocole,
- **Secret :** seulement Alice et Bob peuvent calculer de l'information sur $\boldsymbol{k}$ à la fin du protocole,
- **Intègre :** si Alice et Bob n'abandonnent pas le protocole, ils produisent tous les deux la même clé $\boldsymbol{k}$,
- **Authentifié :** Alice peut être certaine que c'est bien Bob qui a communiqué avec elle pour établir cette clé, et vice-versa.
- **Protection contre le rejeu (Replay Protection) :** Un adversaire ne peut pas rejouer d'anciennes transcriptions du protocole pour établir une clé déjà utilisée,
- **Protection contre la dégradation (Downgrade Protection) :** un adversaire ne peut pas forcer Alice et Bob à utiliser des paramètres cryptographiques plus faibles que ceux qu'ils supportent.
- **Perfect Forward Secrecy (PFS) :** la compromission des clés à long terme d'Alice et Bob (leurs clés de signature) ne permet pas à un adversaire de déchiffrer des sessions passées dont il aurait enregistré les transcriptions.

Votre protocole doit également inclure :

- Une négociation explicite des algorithmes cryptographiques supportés par Alice et Bob,
- Un mécanisme de protection de l'intégrité de cette négociation,
- L'utilisation de valeurs aléatoires (nonces) pour prévenir le rejeu.

Argumenter pourquoi votre protocole satisfait ces propriétés. Pour la propriété **PFS**, expliquer explicitement pourquoi la compromission des clés de signatures ne compromet pas les sessions passées.

---

2. Protocole ''record layer'' avec gestion de sessions multiples

Vous pouvez maintenant supposer qu'Alice et Bob partagent une clé privée $\boldsymbol{k}$ qui est secrète, intègre et authentifiée, établie via le protocole de la question 1.

Hypothèse sur $\boldsymbol{k}$: Notez ici toute hypothèse que vous faites sur $\boldsymbol{k}$ (par exemple, sa longueur, sa structure si vous la dérivez en plusieurs sous-clés, etc.)

Alice et Bob veulent maintenant communiquer de façon bidirectionnelle et gérer plusieurs sessions concurrentes. Donner le pseudo-code d'un protocole qui permet à Alice et Bob d'échanger des messages $\boldsymbol{m_x}$ avec les propriétés suivantes :

- **Robuste :** si les messages transmis ne sont pas modifiés, le protocole ne s'arrête pas prématurément et les messages sont correctement délivrés,
- **Confidentialité :** seulement Alice et Bob peuvent obtenir de l'information des messages m_x échangés,
- **Intègre :** si Alice et Bob n'abandonnent pas le protocole, chaque message reçu est identique au message envoyé,
- **Authentifié :** Le récepteur peut être certain de l'identité de l'émetteur de chaque message,
- **Protection contre le rejeu :** Un adversaire ne peut pas rejouer un ancien message valide et le faire accepter par le récepteur,
- **Protection contre la réorganisation :** Les messages sont reçus dans l'ordre où ils ont été envoyés (ou le récepteur détecte qu'ils sont hors d'ordre),
- **Gestion de sessions multiples :** Alice et Bob peuvent maintenir plusieurs sessions de communication concurrentes (par exemple, Alice peut envoyer des messages à Bob dans différentes « conversations » simultanées), et le protocole doit garantir l'isolation entre ces sessions.

Votre protocole doit inclure :

- Un mécanisme de numérotation de séquence pour chaque message,
- Un identifiant de session pour distinguer les différences sessions,
- Une explication de comment dériver les clés spécifiques à chaque session à partir de k.

Argumenter pourquoi votre protocole satisfait ces propriétés. Expliquer particulièrement comment votre mécanisme de numérotation de séquence et d'identifiant de session prévient les attaques par rejeu et réorganisation.

---

3. Analyse des attaques

Considérez les attaques suivantes contre vos protocoles :

a) Attaque par rejeu sur le handshake

Un adversaire enregistre une transcription complète d'un handshake réussi entre Alice et Bob. Plus tard, il rejoue cette transcription. Expliquer pourquoi votre protocole de la question 1 résiste à cette attaque.

b) Attaque par dégradation (Downgrade attack)

Supposons qu'Alice et Bob supportent tous deux des algorithmes forts (par exemple, AES-256) et des algorithmes plus faibles (par exemple, DES). Un adversaire actif intercepte les messages de négociation et modifie les listes d'algorithmes supportés pour forcer l'utilisation de DES. Expliquer comment votre protocole détecte ou prévient cette attaque.

c) Attaque par rejeu intersessions

Un adversaire enregistre un message chiffré valide $\boldsymbol{c}$ envoyé par Alice à Bob dans la session 1. Il rejoue ce message dans la session 2 (entre les mêmes Alice et Bob). Expliquer pourquoi votre protocole de la question 2 empêche Bob d'accepter ce message dans la session 2.

d) Attaque par compromission de clé à long terme

Un adversaire compromet les clés de signature d'Alice plusieurs mois après qu'Alice et Bob ont eu une conversation. L'adversaire avait enregistré la transcription complète de cette conversation. Expliquer pourquoi (ou pourquoi pas) l'adversaire peut déchiffrer la conversation passée.

---

4. Algorithmes et certificats

a) Avec quel algorithme a été produite pk_C la clé publique de Charlie ?
b) Quelle entrée est-ce que cet algorithme prend ?
c) Est-ce que pk_C est la seule sortie de cet algorithme ? Si non, quelle autre sortie y a-t-il ?
d) Roule-t-il en temps polynomial ? Pourquoi ?
e) Est-ce que cet algorithme pourrait être déterministe ? Pourquoi ?
f) Quel est le contenu exact de $cert_{C\to A}$? Et celui de **$cert_{C\to B}$?**
g) Comment Bob vérifie-t-il la validité du certificat $cert_{C\to A}$ qu'Alice lui envoie durant le handshake ? Donner l'algorithme explicite.

h) La révocation de certificats

Supposons que la clé privée d'Alice soit compromise. Comment Charlie peut-il révoquer le certificat $cert_{C\to A}$? Proposer un mécanisme simple que Bob pourrait utiliser pour vérifier si un certificat a été révoqué avant de l'accepter. Vous n'avez pas besoin d'implémenter ce mécanisme dans vos protocoles, seulement le décrire conceptuellement.

Suggestions :

Bibliographie :
L'utilisation de Zotero (extension dans votre navigateur) afin de créer un catalogue des liens de vos consultations sur le Net. À la fin, cela donne une bibliographie que vous pouvez insérer à la fin de votre PDF. C'est gratuit comme outil et très pratique pour la correction de vos travaux par les auxiliaires d'enseignement.
https://zotero.org

Pseudo-code :
Il existe plusieurs approches et façon de s'y prendre. En voici une utilisée au Collège de Lionel-Groulx, gracieuseté de Vincent Échelard (Coordonnateur académique UdS- programmes d'informatique au Pavillon de Longueuil):
Règles du bon pseudocode