Introduction à la cryptographie

Projet 1 : chiffrement de fichiers

Vous êtes responsable de la conception et du développement d'un utilitaire de chiffrement de fichiers. Cet utilitaire devra être conforme au Référentiel Général de Sécurité version 2 de l'ANSSI et en particulier à son annexe B1 [1]. Il est donc nécessaire de lire tous les chapitres de ce document qui concernent la cryptographie symétrique.

Les mécanismes implémentés dans ce projet doivent se baser sur un algorithme de chiffrement par bloc (les fonctions de hachages sont exclues).

1. L'utilitaire s'utilisera comme une commande en ligne de la forme suivante pour chiffrer ou déchiffrer un fichier :
   `filecrypt –enc|-dec –key K…K –in <input file> -out <output file>`
   La clef est donnée en hexadécimal. Par exemple une clef de 8 bits à 1 sera donnée par –key FF. Le programme doit vérifier que le fichier d'entrée existe et que le fichier de sortie n'existe pas (ou demander la permission explicite de l'écraser).

   Quel algorithme et quelle taille de clef choisissez-vous ? Justifier ce choix (ainsi que les autres dans ce projet) en faisant des références précises au document [1] (numéro de règle ou de paragraphe, citation d'un extrait, etc.) ou à d'autres documents trouvés dans vos recherches si besoin.
   Quel mode d'opération choisissez-vous ? Justifiez.
   Quel *padding* choisissez-vous ? Justifiez.
   En fonction de ces choix, définissez un format de fichier chiffré.

   Attention, afin de pouvoir faire auditer plus facilement votre code source, il vous est demandé de programmer explicitement le mode d'opération de chiffrement et de n'utiliser l'API du langage que pour le chiffrement d'un bloc.

2. Modifiez un fichier chiffré et déchiffrez le. Observez ce qui se passe et justifiez par rapport au mode de chiffrement choisi.

3. Sans changer l'algorithme ni le mode de chiffrement, on décide de rajouter un mécanisme assurant l'intégrité du fichier. Proposer et justifier une solution et un nouveau format de fichier chiffré. Pour cette question il est nécessaire d'effectuer une recherche (hors du RGS) pour trouver comment faire cohabiter le chiffrement et l'intégrité. Implémenter la solution retenue, testez la en modifiant un fichier chiffré.

4. L'utilitaire doit maintenant prendre en entrée une liste de fichiers. La sortie sera une archive compressée sous format zip des fichiers chiffrés. Il est demandé de chiffrer différemment deux fichiers au contenu identique. Vérifier que votre solution est conforme à cette contrainte et expliquez pourquoi.

5. Comparez la taille d'une archive de sortie avec la somme des tailles des fichiers chiffrés. Expliquez ce résultat.

6. Vous apprenez en cours de projet que les fichiers chiffrés doivent être traités par des scripts existants et que ceux-ci sont sensibles à la longueur des fichiers. Il faut donc impérativement produire des fichiers chiffrés de la même longueur que les fichiers clairs (à l'octet près). Par contre vous pouvez rajouter un fichier supplémentaire de gestion du chiffrement dans l'archive produite. Trouvez un nouveau mode de chiffrement répondant à cette nouvelle contrainte (toujours compatible avec le RGS) et modifiez votre format de fichier chiffré / d'archive en conséquence. Expliquez vos choix.

Référence :

[1] Référentiel Général de Sécurité version 2.0 Annexe B1, Mécanismes cryptographiques, Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques. Version 2.03 du 21 février 2014. https://www.ssi.gouv.fr/uploads/2014/11/RGS_v-2-0_B1.pdf

Consignes relatives au projet :

- Les langages de programmation conseillés sont Java et Python mais tout autre langage peut être utilisé.
- Le projet peut être réalisé seul ou en binôme.
- On rendra avant le 29 février 2020 (envoi par mail à pierre.girard@thalesgroup.com) les codes source commentés de chaque étape du projet ainsi qu'un document de conception au format PDF répondant aux questions.
- La notation tiendra compte du raisonnement et de l'argumentation précise des réponses ainsi que de la forme (orthographe, clarté des schémas, etc.). La qualité du code source sera également prise en compte (structure, commentaires, nomage, etc.).