CRY 2023

Laboratoire #2

17-04-2023

Préambule

- Ce laboratoire demande de résoudre différents challenges basés sur la cryptographie symétrique. Chaque étudiant doit rendre un rapport **individuel** avec ses propres solutions ainsi que son code. Les façons d'arriver à la solution sont libres mais doivent être expliquées dans le rapport.
- Vous pouvez par contre discuter ensemble pendant que vous résolvez les problèmes. Essayez de ne pas spoiler !
- Voici une liste de fonctions qui pourraient vous être utiles en python :
  - dans le module pycryptodome [1] : `Crypto.Cipher.AES`, `Crypto.Random` et `Crypto.Util.strxor`.
  - dans le module base64 : `base64.b64encode()`, `base64.b64decode()`
- Toutes les données **binaires** sont en **base64**. N'oubliez pas de les décoder avant de les utiliser.
- Vous trouverez vos entrées personnalisées dans le zip `nom_prenom.zip` sur cyberlearn.
- Le **rapport** et votre **code** doivent être rendus dans un zip sur cyberlearn avant le **01.05 à minuit**.
- Il se peut que j'annonce des erreurs sur cyberlearn/teams.

1 CBC (1.5 pts)

En tant que RH à la banque USB, vous pouvez demander au système de chiffrer d'autres messages de votre choix. Vous avez accès à cette interface via le serveur `iict-mv330-sfa.einet.ad.eivd.ch` sur le port `8000`. Vous aurez besoin du VPN de l'école pour vous y connecter depuis l'extérieur. Ce système est par exemple utilisé pour chiffrer les salaires des employés. Vous pouvez utiliser le code du fichier `ex1_student.py` pour chiffrer des messages.

1. Quel est l'algorithme (précis) utilisé pour chiffrer ces salaires ? N'oubliez pas la taille de la clef. Le code source vous est donné dans le fichier `cbc.py`.
2. Vous avez trouvé le salaire chiffré (`ct`) ainsi que l'IV correspondant d'un dirigeant d'USB (ils sont dans votre fichier de paramètres). Récupérez le salaire ! Expliquez votre attaque en détails. Le paramètre `ID` dans votre fichier de paramètres est l'identifiant du dirigeant dont vous souhaitez connaitre le salaire.
3. Comment pouvez-vous corriger le problème ? Proposez un code n'ayant pas cette vulnérabilité.
4. Est-ce que cette attaque s'applique aussi à AES-CTR ? Justifiez.

2 CCM modifié (2 pts)

Le fichier `ccm.py` contient une implémentation python de CCM développée par la banque USB.

1. Plusieurs éléments sont modifiés par rapport à CCM. Lesquels ?
2. Implémentez le déchiffrement correspondant.

1. https://pycryptodome.readthedocs.io

3. Vous avez intercepté deux textes clairs (m_1 et $m2$) de test ainsi que les textes chiffrés, IVs, et tags correspondants. Utilisez ces informations pour casser la construction. Utilisez vos paramètres pour implémenter votre attaque et donnez le résultat (IV, chiffré, tag) dans votre rapport.
   Indice : Vous allez casser l'intégrité des messages.
4. Corrigez l'implémentation et faites en sorte que le chiffrement CCM soit correct !
   Indice : Ne cherchez pas trop loin.

3 Bruteforce intelligent (1.5 pts)

Un apprenti en cryptographie chez USB décide d'augmenter la sécurité d'AES en ajoutant une couche supplémentaire afin de le rendre résistant aux attaques quantiques. Il décide donc de chaîner deux AES-256 consécutifs avec deux clefs différentes. Plus précisément, un message m est chiffré comme $c = \text{AES}_{k_2}(\text{AES}_{k_1}(m))$.

Le but de cet exercice est de montrer que cette solution n'est pas beaucoup plus sûre qu'un seul AES. Vu que même un seul AES n'est pas cassable en pratique, nous allons **fixer à 0 tous les bits de chaque clefs sauf les 16 premiers**. Du code l'effectuant vous est donné dans le fichier `bruteforce.py`.

1. Vous trouverez dans votre fichier de paramètres un texte clair (`plaintext`) et un texte chiffré (`ciphertext`). Récupérez les deux clefs secrètes (**en base64**) et expliquez comment vous avez procédé.
   Indice : Il peut être utile de stocker des résultats intermédiaires dans un dictionnaire.
2. Décrivez votre algorithme. Quelle est la complexité pire-cas de votre attaque ? Comment se compare-t-elle à un bruteforce des deux clefs ?
3. Qu'est-ce que votre attaque implique sur la complexité du bruteforce sur 3-key 3DES ?

Note : Vos algorithmes ne devraient pas prendre plus de 5 secondes à exécuter.