SAE Crypto - Partie 2

Liaisons épistolaires

Introduction

Alice et Bob vivent un amour passionné mais secret depuis de nombreuses années. La discrétion de leur relation repose en grande partie sur l'échange de messages chiffrés. Pour cela, ils utilisent un protocole nommé *PlutotBonneConfidentialité* qui peut se résumer de la manière suivante:

- A l'aide d'un protocole cryptographique **asymétrique**, ils s'échangent une **clé de session**. Cette clé servira à chiffrer les communications futures.
- Une fois cette clé choisie et échangée, le chiffrement des messages s'opère grâce à un protocole cryptographique **symétrique**. Ils peuvent ainsi communiquer en toute liberté!

Toutefois, Eve, la meilleure amie d'Alice, se doute de quelque chose... Elle voudrait révéler au grand jour leur histoire et va donc tenter de déchiffrer leur communication.

Partie 1: premières tentatives

Informaticienne de formation, Eve commence donc par lire la documentation du protocole et apprend les informations suivantes:

- Le protocole utilise **RSA** pour récupérer la clé de session
- Pour le chiffrement symétrique, une première version consistait à utiliser un algorithme simplifié inspiré de **DES** (SDES)
  - le message est chiffré deux fois avec deux clés différentes: $C = SDES(SDES(M, k_1), k_2)$
  - une implémentation de SDES en python est par exemple fourni à cette adresse: https://jhafranco.com/2012/02/10/simplified-des-implementation-in-python/

Répondez aux questions suivantes:

- En supposant que RSA soit utilisé correctement, Eve peut-elle espérer en venir à bout? En vous appuyant sur votre cours, justifiez votre réponse.
- En quoi l'algorithme SDES est-il peu sécurisé? Vous justifierez votre réponse en analysant le nombre d'essai nécessaire à une méthode "force brute" pour retrouver la clé.
- Est-ce que double SDES est-il vraiment plus sur? Quelle(s) information(s) supplémentaire(s) Eve doit-elle récupérer afin de pouvoir espérer venir à bout du double DES plus rapidement qu'avec un algorithme brutal? Décrivez cette méthode astucieuse et précisez le nombre d'essai pour trouver la clé.

A vous de coder!

- Proposez une méthode `cassage_brutal(message_chiffre, taille_cle1, taille_cle2)` qui tente de retrouver la clé utilisée pour chiffrer le message en testant toutes les possibilités de clé.
- Proposez enfin une fonction `cassage_astucieux(message_chiffre, ?)` qui prend en entrée le message chiffré et potentiellement d'autres paramètres.

Vous testerez vos fonctions sur les exemples fournis sur celene. Vous proposerez également une expérience permettant de mettre en évidence l'écart de temps d'exécution entre le cassage brutal et le cassage astucieux en fonction de la taille des clés. Cette expérience devra figurer dans votre rapport.

Partie 2: Un peu d'aide

A ce stade, Eve a bien travaillé mais se rend compte d'un problème: le protocole a été mis à jour et utilise maintenant l'algorithme AES avec des clés de taille 256 bits.

1. Est-ce vraiment un problème? Justifiez votre réponse.
2. Proposez un protocol expérimental permettant de mettre en évidence la différence de sécurité entre les deux protocoles. Pour cette partie, il est conseillé d'utiliser des modules python spécifiques, comme cryptography ou Pycryptodome. Vous fournirez le code Python permettant de lancer ce protocole expérimentale.

Elle décide de demander de l'aide à son ami MrRobot, célèbre hacker. Par un miracle cryptographique inexplicable (ou une terrible négligence de nos deux amoureux), il arrive à mettre la main sur la clé utilisée par Alice et Bob. Malheureusement, son état mental n'étant pas toujours très stable, il est rare qu'il fournisse une réponse simple mais plutôt des énigmes à résoudre.

Récupérez sur Celene l'énigme proposée, qui vous permettra de trouver la clé. A vous de jouer!

Vous présenterez dans votre rapport non seulement la solution de l'énigme mais également le détail des étapes qui vous ont permis de la résoudre. Si vous avez dû implémenter du code python, il faudra le fournir.

Partie 3: Analyse des messages

La trace de paquets sera disponible prochainement. Un mail vous sera envoyé afin de vous prévenir de sa mise en ligne.

Très fière de son travail de cryptographe, Eve a maintenant en sa possession la clé de session. Elle va pouvoir s'atteler au déchiffrement des messages d'Alice et Bob!

Cependant une dernière difficulté s'impose... Il lui faut récupérer ces échanges et les déchiffre...

Heureusement, MrRobot est plutôt en bonne forme et lui fournit, dans un de ses rares moments de lucidité, une trace d'échanges réseaux contenant notamment certains messages chiffrés grâce au protocole "PlutotBonneConfidentialité". Il conseille également à Eve d'utiliser scapy pour l'aider.

Suivez les conseils de Mr Robot et proposez un programme Python, utilisant scapy, qui vous permette de filtrer les messages reçus afin de ne conserver que ceux d'Alice et Bob, puis déchiffrer les données transportées par ces paquets.

Partie 4: Un peu de recul

Voici quelques questions supplémentaires afin d'aller un peu plus loin dans vos réflexions.

- Alice et Bob utilisent toujours la même clé. Est-ce une bonne pratique?
- Le protocole *PlutotBonneConfidentialité* est inspiré d'un vrai protocole réseau. Lequel? Décrivez la partie associé à la certification des clés qui est absente de *PlutotBonneConfidentialité*.
- Il n'y a pas que pour l'échange de mots doux q'un tel protocole peut se révéler utile... Donnez au moins deux autres exemples de contexte où cela peut se révéler utile.
- Connaissez-vous des applications de messagerie utilisant des mécanismes de chiffrement similaires? (on parle parfois de chiffrement *de bout en bout*)? Citez-en au moins deux et décrivez brièvement les mécanismes cryptographiques sous-jacent.
- Récemment, différents projets de loi et règlements (CSAR, EARN IT Act) visent à inciter voir obliger les fournisseurs de services numériques à pouvoir déchiffrer (et donc analyser) les communications de leur.e.s utilisateur.rices. Discutez des arguments en faveur ou contre ces législations, notamment en matière de vie privée.

Rendu

Vous devrez rédiger un **rapport** rassemblant:

- une page de garde avec les noms, prénoms et groupes des membres du projet
- l'ensemble des réponses aux questions
- votre répartition des tâches

Vous fournirez également l'ensemble du **code** produit pour cette partie. En particulier, vous devez inclure:

- l'ensemble des fichiers `py`
- un fichier `requirements.txt` afin de reproduire l'environnement virtuel dans lequel vous aurez travaillé
- un fichier `README.md` décrivant comment exécuter votre programme.

Le respect des bonnes pratiques de programmation et de gestion de projet vu à l'IUT sont de rigueur. En particulier, votre code devra:

- être versionné avec `git`
- être commenté
- être lisible

Le tout sera rassemblé dans une **archive au format zip** nommée au format:

`nom1_prenom1_gr1_nom2_prenom2_gr2.zip`

L'archive sera à déposer sur Celene avant le **vendredi 16 décembre à minuit.**