UNIVERSITE D'ORLEANS

IUT'ORLÉANS

SAE 3.04
Crypto - Partie 2
Liaisons épistolaires

BRISSET Leo - SEVELLEC Maxime - ÖZOCAK Ibrahim

https://github.com/BRISSETLeo/sae_crypto_sevellec_brisset_ozocak

19/12/2023

Introduction : 3
Tâches Demandées : 4
- Partie 1 : Premières tentatives 4
- Partie 2 : Un peu d'aide 4
- Partie 3 : Analyse des messages : 5
- Partie 4 : Un peu de recul 5
Partie 1: premières tentatives 6
Présentation : 6
1/En supposant que RSA soit utilisé correctement, Eve peut-elle espérer en venir à bout? En vous appuyant sur votre cours, justifiez votre réponse : 6
2/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 : 6
3/Est-ce que le double SDES est-il vraiment plus sûr? 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'essais pour trouver la clé : 7
Partie 2: Un peu d'aide 8
Présentation : 8
1/Est-ce vraiment un problème? Justifiez votre réponse : 8
2/Nous allons tenter d'illustrer expérimentalement les différences entre les deux protocoles. Vous évaluerez: 8
- 2.1/Le temps d'exécution du chiffrement/déchiffrement d'un message avec chacun des deux protocoles : 8
- 2.2/Le temps de cassage d'AES si vous deviez l'exécuter sur votre ordinateur. Ici il faut uniquement estimer le temps nécessaire. Vous préciserez votre configuration et vous fournirez le détail des calculs : 9
3/Il existe d'autres types d'attaques que de tester les différentes possibilités de clés. Lesquelles? Vous donnerez une explication succincte de l'une d'elles : 10
Décryptage d'une Image : 10
Partie 3: Analyse des messages 12
Présentation : 12
Description de la tâche effectuée : 12
Importation des modules : 12
Déchiffrement des messages : 12
Chiffrement des messages : 12
Obtention de la clé : 13
Affichage des messages déchiffrés : 13
Résultat: 13
Partie 4: Un peu de recul 14
Présentation : 14

1/Alice et Bob utilisent toujours la même clé. Est-ce une bonne pratique?.....14
2/Le protocole “PlutotBonneConfidentialité” est inspiré d’un vrai protocole réseau. Lequel? Décrivez la partie associée à la certification des clés qui est absente de “PlutotBonneConfidentialité”.....14
3/Il n’y a pas que pour l’échange de mots doux qu'un tel protocole peut se révéler utile. . . Donnez au moins deux autres exemples de contexte où cela peut se révéler utile.....14
4/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-jacents..... 15
5/Récemment, différents projets de loi et règlements (CSAR, EARN IT Act) visent à inciter voire 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.....15
Répartition des tâches.....16
ÖZOCAK İbrahim :..... 16
BRISSET Leo :..... 16
SEVELLEC Maxime :..... 16

Introduction :

La SAÉ Crypto - Partie 2, intitulée “Liaisons épistolaires”, propose une série de défis complexes sur la cryptographie. Les participants sont invités à explorer et à mettre en pratique leurs connaissances en sécurité informatique en se confrontant à un protocole de communication, le "PlutotBonneConfidentialité". Ce protocole est utilisé par Bob et Alice, deux amoureux qui s’échange des messages chiffrés, il comprend deux étapes majeures : l'échange de clé de session via RSA, un algorithme cryptographique asymétrique, suivi du chiffrement des communications avec un protocole symétrique inspiré de SDES (Simplified Data Encryption Standard).

Tâches Demandées :

Dans cette SAE diverses tâches sont demandées, où chaque membre est responsable de contribuer à ces différentes tâches afin d'assurer le succès de la SAE. Chaque partie propose une suite de tâches à réaliser, cela peut être des questions à répondre ou bien du code à fournir. Voici la liste des tâches à réaliser par partie que nous avons pu relever dans le sujet :

- Partie 1 : Premières tentatives
  - Évaluation d'utilisation du protocole RSA :
    - Analyse sécurité RSA
    - Justification du résultat d'Eve
  - Analyse de SDES :
    - Évaluation de la sécurité SDES
    - Justification en analysant le nombre d'essais nécessaires pour une attaque de force brute
  - Analyse double SDES :
    - Évaluation de la sécurité double SDES
    - Donner des informations supplémentaires pour une méthode casse astucieuse
    - Décrire la méthode casse astucieuse avec le nombre d'essais nécessaires.
  - Extension de SDES :
    - Proposez une extension permettant de coder un texte quelconque, quelle que soit sa taille.
  - Méthodes de Cassage :
    - Méthode cassage_brutal(message_clair, message_chiffre)
    - Méthode cassage_astucieux(message_clair, message_chiffre)
  - Expérimentation :
    - Tester les fonctions sur des exemples fournis sur Celene
    - Proposer une expérience pour évaluer le nombre de tentatives et le temps d'exécution.
- Partie 2 : Un peu d'aide
  - Transition vers AES :
    - Analyse du problème avec des clés de 256 bits
    - Justification
  - Evaluation expérimentale :
    - Mesure expérimentale du temps d'exécution du chiffrement/déchiffrement avec RSA et AES
    - Estimer le temps de cassage d'AES
  - Types d'attaques :
    - Identifier une autre type d'attaque
    - Donner une explication succincte de l'attaque identifier

- Partie 3 : Analyse des messages :
    - Suivre les conseils de MrRobot et proposez un programme Python, utilisant scapy, pour filtrer et déchiffrer les messages
- Partie 4 : Un peu de recul
    - Clé de session :
        - Analyser si l'utilisation de la même clé par Alice et Bob est une bonne pratique.
    - Protocole Inspiré :
        - Identifier le vrai protocole réseau dont “PlutotBonneConfidentialité” est inspiré.
        - Décrire la partie associée à la certification des clés absente dans “PlutotBonneConfidentialité”.
    - Application utiles :
        - Donner au moins deux autres exemples de contextes où “PlutotBonneConfidentialité” pourrait être utile.
    - Applications de messagerie :
        - Identifier au moins 2 applications de messagerie utilisant des mécanismes de chiffrement similaires.
        - Décrire brièvement les mécanismes cryptographiques
    - Législations récentes :
        - Discuter des arguments en faveurs ou contre les législations récentes(CSAR, EARN IT Act) en matière de vie privée.

Partie 1: premières tentatives

Présentation :

La première partie de cette SAE dévoile les premières tentatives d'Eve, pour comprendre et compromettre le protocole en question. À travers l'utilisation de RSA et d'un algorithme symétrique inspiré de DES (SDES), Eve tente de révéler au grand jour l'histoire d'amour entre Bob et Alice, cherchant donc à révéler l'intimité cachée derrière ces messages cryptés. Plusieurs tâches peuvent être distinguées dans cette partie, tout d'abord, il y a 3 questions qui portent sur l'analyse du sujet/protocole de cryptage tel que la possibilité de réussite pour Eve ou bien l'analyse des algorithmes SDES et double SDES. Ensuite, il est demandé de débuter la conception par réaliser une extension permettant de coder un texte quelconque, et de proposer deux fonctions : cassage_brutal(message_clair, message_chiffre) et cassage_astcieux(message_clair, message_chiffre). Enfin, pour conclure cette partie, il faut tester les fonctions réalisées avec des exemples fournis au préalable, mais aussi des expériences permettant d'évaluer les deux fonctions réalisées.

1/En supposant que RSA soit utilisé correctement, Eve peut-elle espérer en venir à bout? En vous appuyant sur votre cours, justifiez votre réponse :

Si l'on considère que RSA a été utilisé correctement, Eve ne peut espérer en venir à bout. En effet, pour décrypter le message, elle aura besoin d'avoir la clé privée. Or, le protocole RSA repose sur la difficulté de factoriser de grands nombres premiers, qui sont nécessaires dans le calcul permettant de trouver la clé privée. En conclusion, cela prendrait donc trop de temps pour trouver les bonnes valeurs.

2/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 :

L'algorithme SDES n'est pas très sécurisé. En effet, il utilise une clé de 10 bits, soit 2^{10} = 1024 clés. Avec un algorithme de type "brute force", cela ne prendrait pas beaucoup de temps pour retrouver la clé.

3/Est-ce que le double SDES est-il vraiment plus sûr? 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'essais pour trouver la clé :

Non, il n'est pas réellement plus sûr, car il est vulnérable à une attaque connue sous le nom de "meet in the middle". Les informations supplémentaires dont l'attaquant a besoin sont le texte en clair "M" ainsi que son équivalent chiffré "C", et il doit connaître la longueur en bits de la clé "K".

Pour réaliser cette méthode astucieuse, l'attaquant collecte toutes les clés possibles en connaissant la longueur de la clé, qui est composée de n bits. Ensuite, il effectue une recherche sur toutes les clés possibles. Dans le cas du double SDES avec des clés de 10 bits, l'attaque MITM nécessiterait en moyenne 2^n = 2^10 = 1024 essais, où n'est la longueur de la clé en bits. Cependant, sachant qu'il y a deux clés, cela prendrait donc 2048 essais.

Partie 2: Un peu d’aide

Présentation :

Dans cette deuxième partie, le protocole a été mis à jour et utilise maintenant l'algorithme AES et des clés de taille de 256 bits. La première étape sera donc l'évaluation de cette transition avec des modules python tels que “cryptography” ou bien “Pycryptodome”, en déterminant si cette évolution présente des problèmes supplémentaires. Ensuite, il faudra réaliser une évaluation expérimentale comparant les temps d'exécution du chiffrement et déchiffrement entre les deux protocoles, et une estimation du temps nécessaire pour casser le chiffrement AES en détaillant la configuration. Par la suite, il est demandé de rechercher d'autres types d'attaques cryptographiques, en donnant pour l’une d’entre elles une explication succincte. Pour conclure, MrRobot fournit deux images d’apparence identique, il va falloir les analyser afin de retrouver la clé tout en présentant la solution et les étapes détaillées, y compris le code Python utilisé.

1/Est-ce vraiment un problème? Justifiez votre réponse :

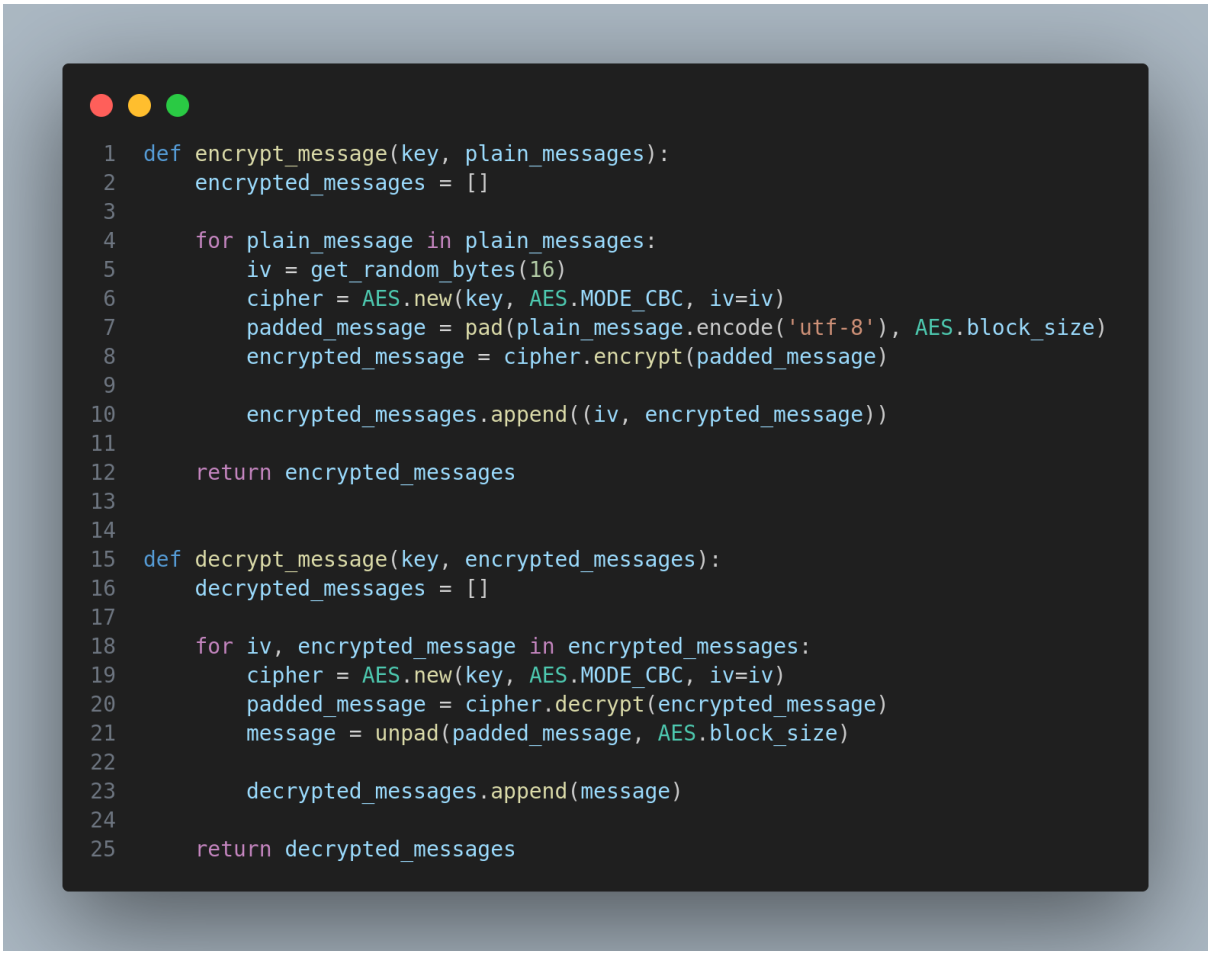
La transition en algorithme AES peut être considérée comme une amélioration significative par rapport à l'algorithme SDES. Cette amélioration est due à la clé AES qui est de 256 bits contre 10 bits pour la clé SDES. De plus, avec un espace de clé plus grand, l'attaque par force brute devient plus difficile, ainsi AES rend les communications plus sécurisées.

2/Nous allons tenter d’illustrer expérimentalement les différences entre les deux protocoles. Vous évaluerez:

- 2.1/Le temps d’exécution du chiffrement/déchiffrement d’un message avec chacun des deux protocoles :

  Voici une capture d'écran du résultat obtenus avec le temps de chiffrement et déchiffrement des protocoles AES et DES ainsi que les fonctions utilisées.
  Ces fonctions sont définies dans le fichier “partie2.py”.

```
Temps cryptage SDES: 0.00002533 seconds
Temps décryptage SDES: 0.00002394 seconds
Temps cryptage AES: 0.00001311 seconds
Temps décryptage AES: 0.00001145 seconds
```



```python
def encrypt_message(key, plain_messages):
    encrypted_messages = []

    for plain_message in plain_messages:
        iv = get_random_bytes(16)
        cipher = AES.new(key, AES.MODE_CBC, iv=iv)
        padded_message = pad(plain_message.encode('utf-8'), AES.block_size)
        encrypted_message = cipher.encrypt(padded_message)

        encrypted_messages.append((iv, encrypted_message))

    return encrypted_messages


def decrypt_message(key, encrypted_messages):
    decrypted_messages = []

    for iv, encrypted_message in encrypted_messages:
        cipher = AES.new(key, AES.MODE_CBC, iv=iv)
        padded_message = cipher.decrypt(encrypted_message)
        message = unpad(padded_message, AES.block_size)

        decrypted_messages.append(message)

    return decrypted_messages
```

- 2.2/Le temps de cassage d'AES si vous deviez l'exécuter sur votre ordinateur. Ici il faut uniquement estimer le temps nécessaire. Vous préciserez votre configuration et vous fournirez le détail des calculs :

  Pour estimer le temps nécessaire à l'exécution de cassage d'AES, nous avons eu besoin de rechercher la complexité de ce protocole avec une clé de 256 bits ainsi que la cadence du CPU de l'ordinateur. La complexité notée X est de 14 d'après divers sources que nous mettrons à la fin de la rédaction, et la cadence du CPU (vitesse maximale du processeur en MHz) note Y égale à 4 600. Nous avons converti Y en Hz puis avons fait le calcul suivant : (X/Y) *12 = 0.00003652, 12 étant égale aux nombres de cœurs disponible dans la machine.

  (14 / 4 600 000) * 12 = 0.00003652 seconde

  voici les sources utilisé pour obtenir la complexité X :

  https://www.websiterating.com/fr/cloud-storage/what-is-aes-256-encryption/

  https://fr.wikipedia.org/wiki/Advanced_Encryption_Standard

3/Il existe d’autres types d’attaques que de tester les différentes possibilités de clés. Lesquelles? Vous donnerez une explication succincte de l’une d’elles :

Il existe en effet plusieurs autres types d’attaques que de tester les différentes possibilités de clés, l’on peut notamment citer les attaques par cryptanalyse différentielle/linéaire, et les attaques par collision.
Voici une explication succincte d’une attaque par cryptanalyse linéaire :
Une attaque cryptanalyse linéaire vise à exploiter les liens linéaires entre les bits d’entrée, de sortie et de clé d’un algorithme de chiffrement. Cette attaque est principalement utilisée contre des algorithmes de chiffrement symétrique tels que DES. Elle va cependant être “inefficace” contre les algorithmes de chiffrement asymétrique tels que RSA, et contre AES qui a été conçue de manière à résister à diverses attaques cryptographiques.
Tout d’abord, à l’aide de cette attaque l'attaquant va commencer par obtenir un ensemble de paires de textes clairs et chiffrés, qu’il va ensuite utiliser pour analyser les relations linéaires entre les bits d’entrée et de sortie de l’algorithme.
Ensuite, après analyse des bits d’entrée, de sortie et clé, l’attaquant créer des matrices de corrélation/liaison linéaire. Avec cette matrice l’attaquant va pouvoir deviner certains bits de clé en fonction des bits d’entrée ou de sortie et des relations linéaire entre les bits révéler par la matrice.
Par la suite, en exploitant les corrélations linéaires identifiées, l’attaquant va encore une fois tenter de deviner des parties de la clé secrète de l’algorithme de chiffrement utilisée. Pour ce faire, il peut utiliser des formules mathématiques afin de déterminer les bits de clé les plus probables.
Enfin, l’attaquant va pouvoir utiliser les informations obtenues pour réduire l’espace de clés possibles, augmentant ainsi les chances de réussite d’une attaque par force brute ou une autre méthode.

Décryptage d’une Image :

Dans cette partie, après avoir récupéré les images fournies sur celene, l’objectif était de résoudre/décrypter l’énigme sur les images. Pour cela l’on a commencé par analyser les bits des images, cependant après de multiple échecs nous n’avons rien découvert,nous nous sommes donc tourné vers les professeurs responsables afin de recevoir de l’aide ou bien quelques indices. Suite à la discussion avec le professeur nous avons appris le positionnement de la clé, sachant que la clé se trouvait dans les derniers bits de l’image 2, on a décidé de ne travailler que sur cette image. Ainsi pour résoudre cette énigme nous avons mise au point et utilisé 4 fonctions : la fonction open_image(path) qui permet d’ouvrir une image selon le chemin donnez, to_binary(image) est fonction permettant de transformer l’image donnez en binaire, block_of_8(lst) cette fonction sert à récupérer pour chaque bloc de 8 bits le dernier bit, enfin last_bit(image) permet de récupérer la clé en transformant les données de l’image en binaire qui va ensuite couper en bloc de 8 bits qui va ensuite récupérer pour chaque bloc le dernier bit, elle va faire cela pour les 512 premiers octets.

(1 octet = 8 bits, taille clé = 64, donc 8 * 64 = 512)
Voici une capture d'écran des fonctions présentées précédemment avec le résultat (fonctions présent dans le fichier “partie2.py”) :

```python
def open_image(path):
    try:
        image = Image.open(path)
        return image
    except Exception as e:
        print(f"An error occurred while opening the image: {e}")
        return None


def to_binary(image):
    try:
        img = Image.eval(image, lambda x: 255 - x)
        data = list(img.getdata())
        data_binary = ''.join(format(255 - pixel, '08b') for pixel in data)
        return data_binary
    except Exception as e:
        print(f"An error occurred while converting to binary: {e}")
        return None


def block_of_8(lst):
    return [lst[i] for i in range(7, len(lst), 8)]


def last_bit(image):
    data_binary = to_binary(image)
    data_bits = block_of_8(data_binary[:512])  # 8 * 64 = 512
    return data_bits
```

```
1110011101101101001100010011111110010010101110011001000001001100
```

Partie 3: Analyse des messages

Présentation :

Désormais, en possession de la clé, le déchiffrement des messages d'Alice et Bob vont pouvoir commencer. Pour cela MrRobot fournit une trace d'échange réseaux contenant certains messages chiffrés à l'aide du protocole "PlutotBonneConfidentialité". Il conseille l'utilisation de "scapy" afin de faire un programme pouvant filtrer les messages et de ne conserver que ceux d'Alice et Bob, pour pouvoir les déchiffrer par la suite.

Description de la tâche effectuée :

Le script Python qu'on a fait à pour but de lire une trace de paquets réseau (fichier pcap) à l'aide de la bibliothèque Scapy, d'extraire des messages chiffrés provenant d'une communication entre Alice et Bob, puis de déchiffrer ces messages à l'aide d'une clé de chiffrement. Le tout est orchestré à partir d'une clé secrète extraite de l'image BMP 'rossignol2.bmp'.

Importation des modules :

On commence par importer les modules nécessaires, notamment AES pour le chiffrement, rdpcap de Scapy pour la manipulation de paquets réseau, et certaines fonctions du module main qu'on a créé pour déchiffrer des messages.

Déchiffrement des messages :

La fonction decrypt_messages prend en entrée une clé de chiffrement et une liste de messages chiffrés (IV + message chiffré).

Elle utilise l'algorithme AES en mode CBC (Cipher Block Chaining) pour déchiffrer chaque message à l'aide de la clé et de l'IV.

Elle effectue le débourrage des messages déchiffrés à l'aide de la fonction pkcs7_unpad pour obtenir les messages originaux.

Chiffrement des messages :

La fonction encrypt_messages prend en entrée le chemin d'une trace de paquets (fichier pcap).

Elle utilise Scapy (rdpcap) pour lire les paquets de la trace.

Elle parcourt chaque paquet, et s'il s'agit d'un paquet IP avec un protocole UDP et un port de destination égal à 9999, elle extrait l'IV et le message chiffré du champ de données UDP.

Ces paires (IV, message chiffré) sont ajoutées à une liste qui est ensuite renvoyée.

Obtention de la clé :

Une image BMP est ouverte à l'aide de la fonction open_image du module main. Cette image est utilisée pour extraire une clé binaire.

La dernière séquence binaire de bits extraite de l'image est répétée deux fois pour créer la clé.

La clé binaire est convertie en une séquence d'octets (bytes) utilisée pour le chiffrement AES.

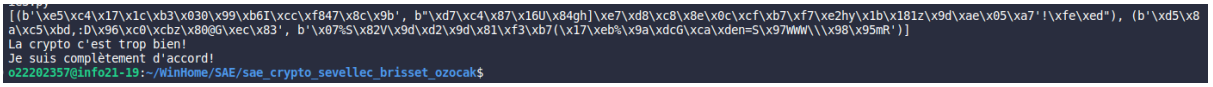
Affichage des messages déchiffrés :

La clé obtenue est utilisée pour déchiffrer les messages extraits de la trace de paquets.

Les messages déchiffrés sont ensuite affichés dans le format UTF-8 à l'aide de la fonction print.

En résumé, le script prend une trace de paquets, extrait les messages chiffrés provenant d'une communication entre Alice et Bob, utilise une clé extraite d'une image pour déchiffrer ces messages, puis affiche les messages déchiffrés.

Résultat:



Partie 4: Un peu de recul

Présentation :

Cette partie 4 permet de conclure la SAE, pour cela, plusieurs questions essentielles sont relevées afin d'approfondir la réflexion. Tout d'abord, l'analyse de la bonne pratique d'Alice et Bob utilisant toujours la même clé. Ensuite, l'analyse du protocole "PlutotBonneConfidentialité" qui semble être inspiré d'un vrai protocole réseau, il va falloir identifier celui-ci et décrire la partie liée à la certification des clés. Par la suite, en élargissant la perspective, ce protocole peut se montrer plus utile dans d'autres concepts, il est donc demandé de relever deux autres contextes où ce protocole va pouvoir s'avérer utile. De plus concernant les applications de messagerie, deux exemples utilisant des mécanismes de chiffrement similaire au chiffrement de bout en bout sont attendus. Pour compléter les exemples il est demandé de fournir des descriptions succinctes des mécanismes cryptographiques sous-jacents. Enfin, pour clore la SAE, il faut argumenter en faveur ou contre les législations telles que CSAR ou EARN UT Act qui incite/oblige les fournisseurs de services numérique à déchiffrer les communications de leur.e.s utilisateur.rices.

1/Alice et Bob utilisent toujours la même clé. Est-ce une bonne pratique?

La réutilisation de la même clé par Alice et Bob n'est pas une bonne pratique en matière de sécurité. L'utilisation d'une clé unique peut augmenter la vulnérabilité aux attaques, en particulier si la clé est compromise à un moment. Il est souvent recommandé de générer de nouvelles clés, afin de renforcer la sécurité du système et de réduire les risques.

2/Le protocole "PlutotBonneConfidentialité" est inspiré d'un vrai protocole réseau. Lequel? Décrivez la partie associée à la certification des clés qui est absente de "PlutotBonneConfidentialité".

Le protocole "Plutôt Bonne Confidentialité" est inspiré d'un protocole appelé "Pretty good Privacy" La signature numérique atteste de la certification. Un utilisateur signe la clé publique d'un autre, cela permet de créer un réseau de confiance entre les utilisateurs.

3/Il n'y a pas que pour l'échange de mots doux qu'un tel protocole peut se révéler utile. . . Donnez au moins deux autres exemples de contexte où cela peut se révéler utile.

En règle générale, un protocole comme celui-ci, qui permet de crypter une communication, peut être utile dans différents cas qui peuvent être plus importants et comporter des enjeux plus significatifs.

On peut, par exemple, imaginer des échanges militaires où les données échangées sont d'une grande importance et ont tout intérêt à ne pas être interceptées et décryptées par d'autres pays.
De même, la communication entre les différents services d'une entreprise, qui ont eux aussi tout intérêt à avoir leurs informations sensibles cryptées.

4/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-jacents.

Il y a par exemple WhatsApp ou Telegram qui utilisent le protocole de chiffrement de bout en bout. Il repose sur l'utilisation de clés publiques et privées pour chaque utilisateur. Lorsqu'un utilisateur envoie un message, celui-ci est chiffré avec la clé publique du destinataire, et seul le destinataire possédant la clé privée correspondante peut le déchiffrer.

5/Récemment, différents projets de loi et règlements (CSAR, EARN IT Act) visent à inciter voire 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.

Le problème principal avec ce projet de loi qui oblige les fournisseurs de services numériques à pouvoir déchiffrer les communications de leurs utilisateurs est que même si cela pourrait permettre de lutter contre la criminalité cela poserait de très sérieuses questions sur la vie privée des utilisateurs en effet leurs communications seront analyser cela peut engendrer une méfiance de la population.

Répartition des tâches

ÖZOCAK İbrahim :

Étant un groupe complet, nous avons tous participé à l'avancement du projet, tous les membres du groupe ont participé pour toutes les tâches relevées. Néanmoins, nous avons quand même désigné un responsable pour chaque partie, designer responsable de la partie 2, ma tâche principale était donc l'avancement de celui-ci. Responsable de la partie 2, j'ai dû faire beaucoup de recherches pour notamment comprendre le protocole AES, pour relever les points positifs de la transition SDES envers AES, mais aussi la recherche de différents types d'attaque. Dans mes tâches responsables, j'ai rencontré des difficultés concernant l'analyse des images fournies, pour surmonter cette difficulté nous nous sommes mis à plusieurs, cependant toujours en difficulté nous en avons discuté avec un autre groupe. D'après le professeur les derniers bits de l'image 2 représentent la clé recherchée, c'est pourquoi nous avons uniquement travaillé sur l'image 2.

BRISSET Leo :

Je me suis occupé de la partie 3 qui consistait à récupérer des messages dans un fichier pcap (fichier utilisé pour stocker des captures de paquets réseau) et j'ai dû pour cela utiliser la clé trouvée dans la partie 2. Une fois la clé trouvé j'ai dû crypter les informations du fichier grâce à Scapy et à la méthode AES et ensuite le décrypter grâce à la clé récupéré pour pouvoir trouver les messages associés à ce fichier de paquets

SEVELLEC Maxime :

Ma partie dans ce projet était d'effectuer la première partie avec le cassage brutal et astucieux ainsi que la réponse à la première question. J'ai bien sûr aussi participé aux autres parties en aidant et réfléchissant avec mes deux camarades. Je n'ai pas eu de difficulté particulière pour réaliser ma partie.
J'ai aussi fait la partie 4 en discutant avec mes coéquipiers, car c'étaient des petites questions qui visaient à nous faire réfléchir et aller plus loin que seulement faire le sujet.