TP 2 : IDEA

Rémy Cassini, Théo Pirkl, Jérôme Chételat
Noria Foukia, Eryk Schiller

Introduction

IDEA pour "International Data Encryption Algorithm" est un algorithme de chiffrement symétrique qui a été développé à l'ETHZ (École Polytechnique Fédérale de Zürich) en 1990 - 91. C'est un algorithme qui utilise des opérations d'arithmétique modulaire telles que celles que vous avez déjà implémenté dans la première partie du TP1.

Un papier scientifique décrivant IDEA et surtout une version simplifiée d'IDEA vous est fournie et vous donne en détail le fonctionnement de cet algorithme. Vous implémenterez uniquement la version simplifiée en cours, mais cela vous permettra déjà de comprendre le fonctionnement de ce chiffrement.

Environnement

Vous utiliserez Python ainsi que votre éditeur de texte ou IDE préféré. Nous vous fournissons dans une archive *.zip* deux fichiers nommés : `idea.py` et `idea_tests.py`.

Le premier, `idea.py`, contient un squelette avec quelques définitions de fonctions que vous aurez à coder pour IDEA. Vous aurez certainement à créer d'autres fonctions vous-mêmes. Ne changez pas le nom des fonctions déjà présentes dans le squelette !

Le second, `idea_tests.py`, est une batterie de tests à utiliser pour vérifier que votre code fonctionne correctement. Ne modifiez **rien** dedans !

Le dernier, `IDEA.pdf`, est un papier scientifique écrit en anglais décrivant l'algorithme IDEA. Attention, il y a une erreur au tout début de la page 4, il manque 8 bits à la clef fournie. (Elle est écrite juste avant, à la fin de la page 3, pour référence.)

La commande pour lancer les tests est :
```
python3 -m unittest idea_tests.IDEATestCase --verbose
```

Évaluation

Ce TP durera **deux séances**. Il est **évalué** et devra donc figurer dans votre journal de laboratoire. Le code compte pour **2/3** et le journal pour **1/3**.

Votre code doit être un minimum commenté et vous devez être capable de l'expliquer à l'oral si nous avons des doutes sur son origine.

Nous vous rappelons qu'il est **formellement interdit** d'utiliser une IA générative (ChatGPT et autres) pour générer votre code.

Exercices

Vous allez implémenter votre propre algorithme IDEA simplifié en suivant les explications données à l'oral et le papier scientifique décrivant IDEA.

Avant toute chose, reprennez votre **TP1** et effectuez la partie 2 si ce n'est pas déjà fait. Revenez ici une fois que vous aurez terminé.

Vous allez commencer par implémenter le chiffrement (car le déchiffrement est identique, à l'exception de la clef que vous devrez manipuler).

Assurez-vous que votre fichier `ssi_lib.py` (ou une copie de celui-ci) est bien dans le même répertoire que votre fichier `idea.py`, car vous devez importer les fonctions utiles du premier dans le second.

Dans les grandes lignes, votre code doit suivre les étapes suivantes :

- Demander à l'utilisateur la clef et le message (en binaire) ainsi que le mode (encoder ou décoder), selon ce qui est stipulé dans `idea.py` initialement (Argparse, pas de `input()`) ;
- Générer toutes les sous-clefs à partir de la clef que l'utilisateur a fournie ;
- (Uniquement pour le déchiffrement), transformer les sous-clefs pour le déchiffrement ;
- Faire autant de rounds que nécessaire pour IDEA simplifié ;
- Faire le dernier demi-round ;
- Afficher le résultat (en binaire).

Attention ! Lors de l'opération de multiplication modulaire, on travaille dans $\mathbb{Z}_{17}^*$, c'est-à-dire les nombres entiers de 1 à 16 inclus. Or, sur 4 bits, vous ne pouvez coder que les nombres de 0 à 15 inclus.

Pour pallier ce problème, lorsque votre multiplication prend un 0 comme l'un des facteurs, remplacez-le par 16 avant d'effectuer l'opération. Exemple :

$$mul\_mod(\mathbf{0}, 3, 17) = \mathbf{16} \cdot 3 \mod 17 = 14$$

De même, à la fin de votre multiplication modulaire, si vous obtenez 16, il faudra le retransformer en 0, car 16 nécessiterait 5 bits pour être encodé, mais vous n'avez le droit qu'à 4. Exemple :

$$mul\_mod(4, 4, 17) = 4 \cdot 4 \mod 17 = \mathbf{16} = \mathbf{0}$$

L'explication se trouve à la page 3 du papier sur IDEA, dans le paragraphe 4.

Pour aller plus loin...

Comme vous l'avez sûrement remarqué, avec votre IDEA simplifié, vous ne pouvez uniquement chiffrer que des messages d'une longueur de 16 bits. C'est peu pratique pour envoyer des messages...

1. Combien de caractères (en ASCII étendu) peut-on chiffrer à la fois avec IDEA ?
2. Comment procéderiez-vous pour chiffrer le message "COUCOU" avec IDEA ? (Cherchez le concept de chiffrement par bloc.)
3. Maintenant, on veut chiffrer le message "SALUT", quel est le problème ?
4. Comment procéderiez-vous pour remédier à ce problème ? (Cherchez le concept de *padding*.)

Implémentez ces changements dans votre code, afin de pouvoir chiffrer des messages d'une longueur arbitraire (toujours en binaire cependant).

Si vous avez bien implémenté votre algorithme et que tous vos tests passent, essayez maintenant d'échanger un message chiffré avec un autre groupe.