Projet PX222 maths/info:
Implémentations d'AES en Haskell et en C.

Grenoble INP – Esisar – année 2022-23

Introduction

La sécurité des données échangées en ligne s'appuie sur les produits d'une science assez fondamentale, située à l'intersection des mathématiques et de l'informatique: la *cryptographie*.

Héritière moderne des codes secrets des anciens, elle va plus loin que la simple protection de la confidentialité des données: authentification des personnes, intégrité des données, authentification des messages, sont quelques autres services qu'elle nous rend au quotidien.

Dans ce projet, nous vous proposons de vous frotter à l'implémenation logicielle d'une *primitive cryptographique*, l'algorithme de chiffrement *AES*.

AES - pour *Advanced Encryption Standard* - est le nom d'un standard définissant un algorithme de chiffrement, promulgué en l'an 2000, visant à remplacer le précédent standard, *DES* - pour *Data Encryption Standard*. DES avant été promulgué en 1977, et il était largement considéré comme périmé.

L'algorithme standardisé par le NIST sous le nom d'AES est en fait l'un des candidats à une compétition organisée par le NIST (National Institute of Standards and Technology), visant à élire le meilleur algorithme de chiffrement à fin de standardisation. Le gagnant est le candidat *Rijndael*, algorithme mieux connu aujourd'hui sous le nom d'AES.

Suite à la standardisation d'AES, le document FIPS 197 annonçant le standard est le document qui fait référence pour la description de cet algorithme. FIPS signifie: Federal Information Processing Standards; c'est une série de standarsd émis par le NIST, au sujet des procédés de traitment de l'information. Vous le trouverez le texte du standard FIPS 197 à l'adresse suivante: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf

Objectifs pédagogiques

Le projet qui vous est proposé est un projet pluridisciplinaire: il vous faudra utiliser votre bonne compréhension des structures algébriques, ainsi que votre savoir-faire en informatique, à la fois en Haskell et en C, pour arriver au bout du projet.

C'est un projet fort en crédits ECTS: 2,8 crédits ECTS, soit entre 70 et 100 heures de travail pour chaque étudiant. C'est bien plus que le temps passé en séance: ce projet vous met donc au défi du *travail en autonomie*.

C'est aussi une réalisation d'envergure, qui ne peut être complétée qu'en y revenant à de nombreuses reprises. Il sollicitera donc votre *capacité à vous organiser*.

Enfin, le temps passé sur le projet ne sera réellement productif que si vous trouvez les *bonnes méthodes de travail*.

Nous sommes là pour vous accompagner; et espérons que ce projet sera pour vous une expérience enrichissante.

Description globale du projet

Il vous est proposé de réaliser un programme informatique de taille moyenne, mais de complexité de mise en oeuvre conséquente. Le code final visé est une implémentation en C d'AES, éventuellement accompagnée d'une application utilisant le chiffrement.

La méthode à appliquer est l'appropriation de l'algorithme via la lecture du document FIPS 197; l'implémentation en Haskell; puis l'implémentation en C.

Parmi les retombées disciplinaires attendues, nous pouvons mentionner:

- découverte du monde de la cryptographie;
- mise en pratique des connaissances théoriques acquises dans le domaine de l'algèbre;
- connaissance approfondie d'un système de chiffrement de bloc (AES);
- pratique avancée du langage C;
- encodage de structures de données en langage C;
- pratique avancée du langage Haskell;
- encodage de structures algébriques en Haskell via le mécanisme de classes, et la définition de types équivalents;
- réalisation et validation d'un code complexe en C;
- méthodes de développement incrémentales et tests;
- utilisation d'un langage de plus haut niveau que C (ici Haskell) comme moyen de prototypage d'un programme à réaliser en C.

Maintenant que vous avez l'eau à la bouche, penchons-nous sur le calendrier du projet.

Calendrier

Le projet se déroulera en deux temps: 'semestre' et 'semaine finale'.

Au cours du second semestre, vous aurez 5 séances de 3h consacrées au projet maths/info. C'est bien moins que le temps nécessaire pour arriver au jalon qui vous sera proposé pour la fin de cette période. Il vous faudra donc travailler régulièrement entre les séances pour arriver à votre objectif.

Ce jalon, c'est votre implémentation en Haskell d'AES et des structures algébriques sous-jacentes. Il sera rendu accompagné du carnet de bord, ainsi que d'une brève documentation illustrant ses fonctionnalités, et les choix techniques effectués;

La date de remise de ce rendu intermédiaire est *au plus* 7 jours après la 5^ème et dernière séance du semestre, à minuit.

Bien que la date de remise soit après la 5^ème séance du semestre, il vous est recommandé de suivre le calendrier de travail suivant:

- séances 1, 2, 3: travail limité à l'appropriation d'AES, et au code Haskell;
- séances 4, 5: début des travaux sur le code C; fin du travail sur l'implémentation Haskell.
- séances de la semaine finale: finalisation du code C.

Lors de la semaine bloquée, vous aurez quatre jours pleins pour finir votre projet; le dernier jour sera consacré aux évaluations finales. Il n'est pas raisonnable d'espérer travailler entre les séances lors de cette semaine.

Méthodes de travail

Les séances de projet sont un temps précieux dont il vous faudra tirer un profit maximum pour l'avancement de votre projet. La présence aux séances est obligatoire. Le travail réalisé entre les séances sera à présenter en début de séance, et sera évalué.

Pour garder la trace des différentes étapes de votre travail, et pour faciliter le partage de l'information au sein de votre binôme, nous vous demandons de tenir à jour un *carnet de bord*.

Le carnet de bord consistera en un unique fichier (au format markdown) progressivement enrichi au fil de l'avancement du projet. A chaque nouvelle séance de travail (présentielle ou entre les séances semestre), vous le remplirez avec la date du jour, l'état d'avancement du projet, vos choix de conceptions, les tâches entreprises, et les éventuelles sources utilisées.

Il servira aussi de point de départ au dialogue avec l'équipe enseignante pour l'évaluation du travail individuel entre les séances de la phase 'semestre' :

- à l'issue de chacune des 4 premières séances de cette phase, vous y mentionnerez pour chaque membre du binôme les tâches à faire pour la séance suivante;
- préalablement à chacune des 4 dernières séances (toujours de la phase semestre), vous y mentionnerez (à nouveau pour chaque membre du binôme) les tâches réellement accomplies depuis la séance précédente.

Pour mener à bien le projet, il sera crucial d'identifier des (petits) modules à réaliser et à tester intensivement (tests archivés et rejouables), avant de les assembler pour réaliser des fonctionnalités plus complexes.

Évaluation

La note finale pour le projet maths/info sera le résultat de quatre évaluations.

L'assiduité, l'attitude et le travail en séance seront évalués individuellement et compteront pour 10%. Le travail entre les séances de la phase 'semestre' est fondamental. il sera évalué lors d'un échange entre chaque binôme et l'équipe enseignante à chacune des 4 dernières séances de la phase semestre. Ce qui aura été consigné dans le carnet de bord servira de point de départ au dialogue. Cette partie donnera lieu à une évaluation inviduelle comptant pour 40% du total.

Le jalon intermédiaire est rendu par des moyens électroniques, et devra comporter:

- le code source;
- le carnet de bord;
- la documentation: description des fonctionnalités et des choix techniques effectués.

L'évaluation finale reposera sur des présentations:

- du code source;
- du carnet de bord;
- des tests des programmes réalisés (automatiques et/ou interactifs).

L'évaluation du jalon intermédiaire comptera pour 20%, et l'évaluation finale, pour 30% de la note finale. La qualité de la démarche scientifique et expérimentale adoptée, et la construction solide d'étapes intermédiaires validées par test formeront une part importante de l'évaluation.

Nous rappelons également le fait que la copie de code, partielle ou totale, entre les binômes ou depuis Internet, constitue du plagiat et sera sanctionnée comme telle.

Il est possible de discuter, échanger, s'inspirer; mais **il est interdit de copier**.

Travail à faire

AES est un algorithme de chiffrement par bloc, chiffrant des blos de données de 128 bits, et utilisant pour cela une clé, qui peut être de 128, 196 ou 256 bits.

AES a été conçu à partir de certaines structures algébriques, que vous reconnaîtrez facilement en lisant le standard définissant *AES*: corps, anneaux, quotients, …

Plutôt que d'aller directement vers une implémentation d'AES en C, nous vous demandons de commencer par implémenter non seulement AES, mais toutes les structures et constructions algébriques sous-jacentes, en Haskell.

Implémenter ces mêmes structures en C serait réellement difficile, et peu commode. Les implémenter en Haskell est une tâche réalisable.

Nous vous demandons une implémentation complète des structures algébriques sous-jacentes à AES. Une fois ces structures implémentées, coder AES en Haskell est plus facile; cela reste une tâche d'une certaine ampleur.

Lorsque vous aurez une implémentation fonctionnelle d'AES en Haskell, celle-ci vous servira à mettre au point votre implémentation d'AES en C.

AES étant une construction assez élaborée, cela vous prendra du temps d'en faire le tour; de vous l'approprier; de l'implémenter. Profitez du fait qu'Haskell vous permette d'écrire des calculs assez abstraits pour vous approprier de proche en proche AES et ses structures: avancer dans le code Haskell vous permettra d'avancer dans la compréhension du standard FIPS 197.

Quelques suggestions pour bien démarrer

Il vous faudra bien évidemment vous procurer, et vous approprier, le document FIPS 197. L'usage de surligneurs, de papier et de stylos est recommandé pour cette phase du projet.

Nous vous donnons également deux fichiers Haskell, qui ne vous serviront peut-être pas au sein d'une implémentation d'AES, mais illustrent une discipline permettant d'encoder facilement des structures algébriques en Haskell.

La logique est la suivante:

- une catégorie de structure algébrique (groupe, anneau, corps, etc) est représentée par une classe, qui encode les 'éléments' de la structure: par exemple, *neutre*, *inverse* et *opération* dans le cas des groupes;
- une structure algébrique concrète est représentée par un type de Haskell; le type est à instancier dans la classe adéquate;
- comme Haskell ne vous laisse faire qu'une seule déclaration d'instance par type, vous pouvez "dupliquer" un type à l'aide du mot-clé `newtype`. `newtype` s'emploie exactement comme `data`, mais avec les restrictions suivantes: il ne permet qu'un seul constructeur, embarquant un unique type comme paramètre.

Tout ceci est illustré dans les fichiers fournis. Il est fortement conseillé de prendre en main (dessins, tests personnels) ce code avant d'écrire la moindre ligne de "vrai code".

Quelques mots de conclusion

Vous avez peut-être trouvé cette lecture un peu longue: c'est un signe qu'il faudra la reprendre: peut-être plusieurs fois; peut-être avec à la main de quoi noter ou annoter.

Ce document sera amené à être complété par d'autres documents, en particulier concernant l'usage de `git` pour la collaboration, et l'archivage des sources et du carnet de bord. Cet usage de `git` sera aussi un moyen pour nous de visualiser votre avancement et le travail invididuel de chacun · e. Le carnet de bord sera présent et versionné le plus possible.

De plus, au fur et à mesure de l'avancement du projet, nous préciserons probablement l'énoncé et probablement quelques "mini-jalons" intermédiaires.

Nous vous souhaitons un excellent projet maths/info, plein d'apprentissages et de réussites; et n'oubliez pas: si vous avez une difficulté, n'hésitez pas à nous solliciter, en ou hors séance: nous

ne ferons jamais le travail à votre place, mais nous ferons du mieux que nous pouvons pour vous aider à avancer.

Laure Gonnord, Vincent Guisse, Yann Kieffer