Documentation générale du package Kruptor_2.1

1 Le logiciel xcas

Le logiciel Xcas est une interface de Giac, une bibliothèque C++ de calcul formel libre (licence GPL). Il est développé par Bernard Parisse de l'Université Joseph Fourier de Grenoble. On peut trouver toutes les informations nécessaires sur Xcas sur le site :

http ://www-fourier.ujf-grenoble.fr/∼parisse/giac_fr.html

Ce logiciel est très utile pour diverses choses, et en particulier pour montrer des fonctionnements d'objets scientifiques divers. Pour la cryptographie par exemple, on peut effectivement programmer des primitives en vraie grandeur, détailler leur fonctionnement, comparer leurs avantages, voir leurs faiblesses etc. La version de xcas au moment de l'écriture de Kruptor_2.1 (Mars 2016) est la version 1.1.1-12

2 Le package Kruptor

Kruptor est un ensemble de fonctions écrites pour Xcas dans le mode Maple. Ces procédures sont à usage pédagogique, et servent à illustrer des cours de cryptographie, ou de boîte à outils pour la réalisation de travaux pratiques. Elles peuvent aussi servir à écrire des tests ou mettre en place des simulations. La version actuelle est la version 2.1. Elle remplace la version 2.0 devenue obsolète car certaines fonctions ne se compilaient plus correctement avec la version actuelle de xcas.

2.1 Auteur

La diffusion de ce package est assurée par l'association ACrypTA, son auteur est Ainigmatias Cruptos en personne. Le site de l'association ACrypTA est :

http ://www.acrypta.com

2.2 Composition du package

Le package est composé de 2 répertoires :

- **commun** : ce répertoire contient **les** fichiers, bibliothèque de fonctions, relatif aux procédures **kruptor_commun_2.1.map, kruptor_rsa_2.1.map, kruptor_elgamal_2.1.map, kruptor_hash_2.1.map**.
- **Documentation** : ce répertoire contient toute la documentation, à savoir la documentation sur le fonctionnement général, la documentation sur chacune des bibliothèques.

Le tout est archivé et compressé dans un fichier (au choix) appelé **Kruptor_2.1.tgz** (ou **Kruptor_2.1.zip**) qui est **le seul fichier à télécharger**.

3 Utilisation

3.1 installation

Pour installer le package, il faut le désarchiver (le désarchivage crée un répertoire kruptor_2.1 qui contient les 2 sous-répertoires "commun, documentation").

3.2 Utiliser les divers bibliothèques

Celles-ci se trouvent dans le répertoire "commun". Il suffit de charger la bibliothèque des procédures communes et les bibliothèques dont on a besoin. Ces bibliothèques se trouvent dans le sous-répertoire "commun". Pour cela voir la documentation de xcas, ou alors procéder comme suit 1) lancement de xcas, 2) tapper "Alt P" 3) il apparaît une nouvelle barre surmontant une fenêtre avec les items "Prog, Edit, Add, nxt, OK, save" au dessous de la ligne "Fich, Edit, Save". 4) Dérouler le menu sous Prog (de cette nouvelle ligne) et choisissez "insert" puis "file". 5) charger le fichier bibliothèque voulu. 6) le chargement s'effectue, appuyez alors sur "OK". 7) Si vous avez d'autres bibliothèques à charger revenez au 4).

Attention, si on veut utiliser les fonctions qui se trouvent dans kruptor_hash_2.1.map, il faut avant de charger ces fonctions avoir lancé pari() dans xcas. En outre la fonction kdf(state,j) de ce composant fait appel au paramètre d'entrée state qu'il faut initialiser avec un germe. Ceci peut être fait par la commande read("seed") dans xcas (le fichier seed qui définit une variable "germe" est fourni, mais peut être modifié à volonté). On appelle alors la fonction kdf() avec pour premier paramètre la variable germe et pour deuxième paramètre le nombre d'octets voulus.

3.3 Les limites

3.3.1 Choix des fonctions

Cette bibliothèque est une petite bibliothèque à usage pédagogique et démonstratif. Donc elle n'a pas prétention à être exhaustive, loin de là. Les procédures développées le sont en fonction des besoins pour l'illustration de certaines parties du cours et du temps libre de l'auteur.

3.3.2 limites de rapidité

Si les systèmes généraux de calcul formel ont l'avantage de la simplicité de mise en œuvre, en revanche ils sont très lents, surtout lorsqu'on développe des fonctions qui ne sont pas natives. Donc les temps d'exécution des procédures proposées n'ont rien à voir avec ceux qu'on peut atteindre avec un langage comme le langage C, voire en assembleur. De ce fait des primitives comme sha256 deviennent ici très lentes et inapplicables à de très longues chaînes de caractères.

3.3.3 Les procédures ne sont pas blindées

Les procédures programmées ne sont pas blindées contre une utilisation erronée. C'est-à-dire qu'on suppose pour chacune d'entre elles que les entrées sont dans le domaine imposé. Si une procédure suppose qu'une entrée x est un entier dans un intervalle donné, et si on l'applique à un entier hors de cet intervalle, tout peut arriver, y compris une exécution infinie. Il ne faut pas oublier qu' un tel système s'utilise de manière interactive. On peut à tout moment lancer des commandes à la ligne de commande si on veut préciser un résultat, ou tester un comportement.

3.3.4 Niveau des fonctions décrites

La plupart des primitives programmées, sont des primitives du niveau primitives mathématiques et ne prennent donc pas en compte les standards plus élaborés de chiffrement ou de signature qui comportent des prétraitements des messages, des formats d'entrée et de sortie normalisés etc.