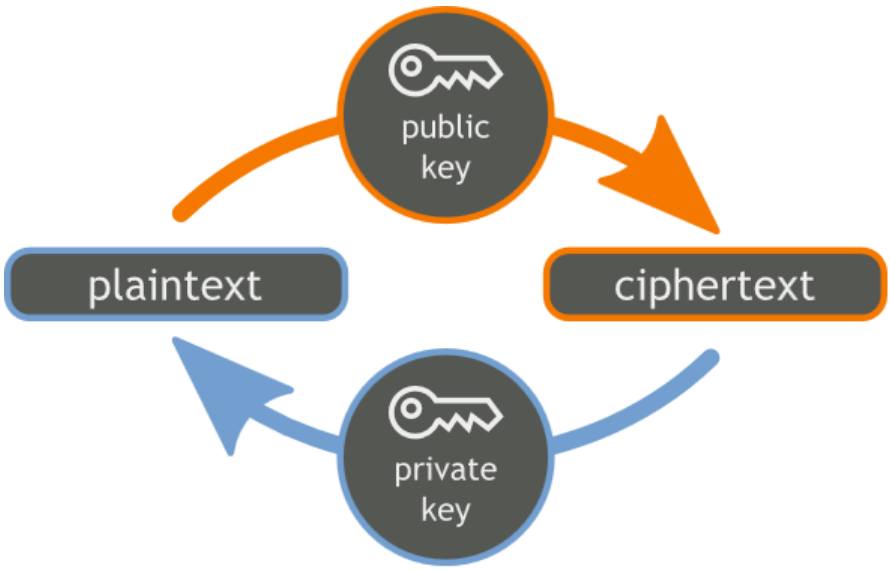
Introduction au langage C (NF05)
— **projet** 2018-2019 —

Taha Arbaoui, Florian Blachère & Rémi Cogranne)
Université de technologie de Troyes

Outil de chiffrement / déchiffrement RSA



Consignes

- Travail en monôme ou en binôme.
- Rapport (en pdf) + code + fichier exécutable sont à envoyer, en un seul fichier zip, à `taha.arbaoui@utt.fr`, `remi.cogranne@utt.fr` et `florian.blachere@utt.fr`, au plus tard le 21 décembre 2018, avant minuit.
- La présentation (power point + exécution) se fera pendant la dernière séance de TD après les vacances de Noël 2018.
- Le code devra être commenté en entier. Pour cela, il faut utiliser l'outil Doxygen [1] pour générer la documentation.
- Recommandations : le langage C est très renseigné sur internet, faites une recherche avant de contacter un ami ou un prof.
- Il est interdit de reprendre un code de quelqu'un autre au risque d'une sanction.
- Il est crucial de citer vos références.

Rapport

Le rapport doit inclure :

1. une introduction qui énonce clairement le problème, ainsi que le plan du document
2. une partie qui décrit les algorithmes utilisés (le fonctionnement et non pas le code)
3. les problèmes rencontrés et les solutions que vous avez adoptées
4. un mode d'emploi du programme
5. une conclusion et des perspectives pour améliorer votre programme
6. un annexe qui comprend le code commenté

Il est également demandé d'estimer la durée de réalisation des différentes parties du code.

Travail demandé

Le but de ce projet est de proposer un outil pour chiffrer et déchiffrer des messages en utilisant le cryptosystème asymétrique (ou à clé publique) RSA.
La description de l'algorithme est la suivante :

- Le premier personnage (Alice) choisit deux nombres entiers premiers grands p et q.
- Alice calcule $n = p \times q$ et $\Phi(n) = (p-1) \times (q-1)$.
- Alice choisit aléatoirement un nombre $e < \Phi(n)$ tel que $PGCD(e, \Phi(n)) = 1$, c'est dire e ne doit pas être un diviseur de $\Phi(n)$ (on pourra vérifier que le reste de la division Euclidienne de $\Phi(n)$ par e ne donne pas 0).

1. `www.doxygen.org`

— Enfin, Alice calcule d tel que $e \times d \equiv 1 \bmod \Phi(n)$.
ATTENTION : erreur dans l'énoncé jusqu'au 8/10 : il faut bien lire $e \times d \equiv 1 \bmod \Phi(n)$ et non pas $e \times d \equiv 1 \bmod n$ comme cela été indiqué auparavant.

Alice publie la clé "publique" (e, n) et garde d, la clé secrète, pour elle.

Avec la clé (e, n), n'importe quelle personne peut envoyer un message chiffrer à Alice. Soit m un message (m est ici un nombre), le message chiffré, noté c est donné par :

$$c = m^e \bmod n$$

Alice déchiffre les messages qui lui sont envoyés en calculant :

$$m = c^d \bmod n$$

Le but de ce projet est de proposer un outil permettant de chiffrer et déchiffrer des messages avec RSA. La principale difficulté est qu'en cryptographie, les nombres manipulés $p, q, n, m, c, d \ldots$ sont très grand.
Vous devrez donc gérer les entiers comme des tableaux. Par exemple l'entier 789456123 devra être représenté dans votre code sous la forme d'un tableau ...
Vous devrez également écrire les codes nécessaires pour réaliser des additions, multiplication et puissance d'entier (pour la puissance, implémentez l'algorithme d'exponentiation modulaire / rapide). Vous devrez également implémenter l'algorithme d'Euclide étendu ou utiliser le théorème d'Euler pour calculer d à partir de e.

Votre projet doit permettre :

— de choisir deux entiers p et q à partir desquels les clés publiques (e, n) et privées (d) doivent être générées automatiquement.
— de choisir un fichier et de le lire par blocs de $B = \lfloor log_2(n) \rfloor$ bits.
— de chiffrer le fichier en utilisant la clé publique.
— de déchiffrer le document en utilisant la clé secrète.

Voici un exemple des opérations que votre programme doit pouvoir faire :

```
Entrez un entier
7894561321378945613213789456132137894561321378945613213789456132137894561321378945613
213789456132137894561321378945613213
Longeur : 329 bits

Entrez un entier
7984649816519846516549768135186494
Longeur : 113 bits
n=p*q :
```

```
630353076062530751594288705981208138005357303380053573033800535
730338005357303380053573033800535729707652281240849301978745094
5545222
longueur : 442 bits

Entrez une cle de chiffrement (publique)
17
longueur :   5 bits
Calcul de la cle de dechiffrement ..........
Cle de chiffrement et de dechiffrement ecrite dans les fichiers
cle_privee.key et cle_publique.key

Entrez le nom du fichier a chiffrer
NB : Le fichier sera lu par block de 441 bits
NB : Des '0' seront ajoutés en fin de fichier pour completer le
dernier bloc.
poeme.txt

Entrez le nom du fichier contenant la cle publique
cle_publique.key

Chiffrement enregistre dans le fichier texte_chiffre.txt
```

Bonus

Un bonus de points est donné si l'application permet aussi de :

— Trouver automatiquement des nombres premiers p et q (utiliser l'algorithme de Rabbin-Miller).

— d'optimiser le déchiffrement en utilisant le standard PKCS#1 et le théorème chinois des restes.

— toutes les améliorations possibles sont les bienvenues !