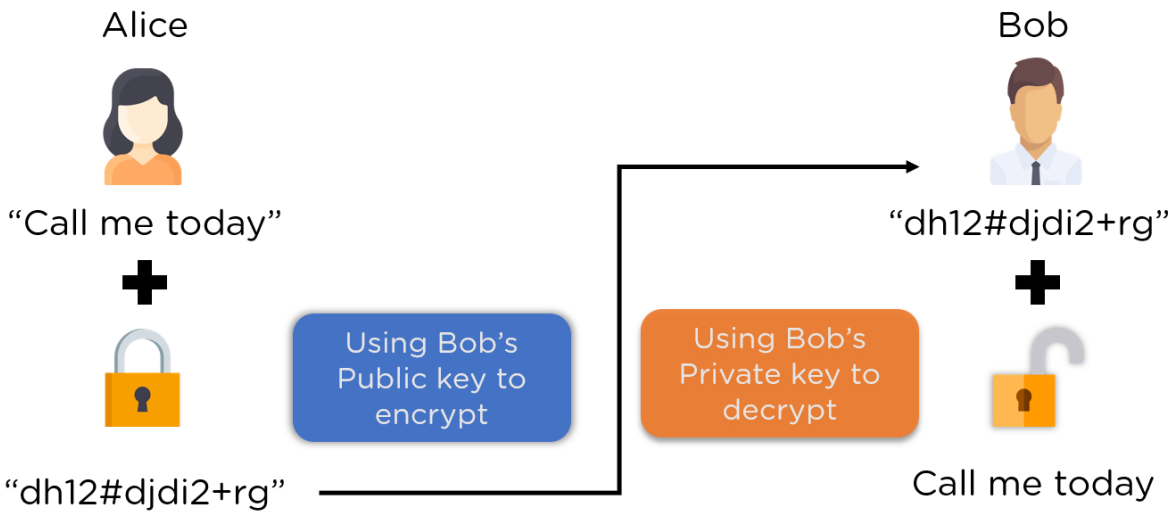
BENKHALED ZINEDDINE 191931047529

RSA (Ronald Rivest, Adi Shamir, Leonard Adleman) RAPPORT



Introduction

RSA est algorithme de cryptage asymétrique basé sur deux clés une publique utilisé pour le chiffrement et une clé privé pour le déchiffrement. Il chiffrement se base sur le calcul du modulo et le déchiffrement sur le calcul de l’inverse module. De manière générale, il se base sur l’exponentiation modulaire.

Analyse de l’algorithm

Structure de données utilisées

- **bignum:** structure de données représentant un entier de taille arbitraire.
- **bignum *encoded:** tableau de bignum représentant le message encodé
- **int *decoded:** tableau d'entier représentant le message décodé
- **char *buffer:** tableau de caractère représentant le message lu dans le fichier

- **FILE *f**: pointeur sur le fichier lu "message_M.txt" pour lire le message a crypté.

Paramétres clés des algorithmes

- **bignum *p:** premier facteur du module
- **bignum *q:** second facteur du module
- **bignum *n:** module
- **bignum *phi:** totient
- **bignum *e:** exposant public
- **bignum *d:** exposant privé
- **bignum *bbytes:** nombre de caractères encodés dans un bloc de chiffrement

Complexité temporelle

Nous discutons d'abord de la complexité des fonctions utilisées dans le code

- **bignum_fromint:** O(1)
- **bignum_frombytes:** O(n)
- **bignum_fromstr:** O(n)
- **bignum_frombignum:** O(n)
- **bignum_tostr:** O(n)
- **bignum_tobytes:** O(n)
- **bignum_print:** O(n)
- **bignum_deinit:** O(1)
- **bignum_add:** O(n)
- **bignum_sub:** O(n)
- **bignum_imultiply:** O(n^2)
- **bignum_idivide:** O(n^2)
- **bignum_imod:** O(n^2)
- **bignum_imodinv:** O(n^3)
- **bignum_less:** O(n)

n = le nombre de caracteres de message.

Complexité spatiale

- **bignum_fromint:** O(1)
- **bignum_frombytes:** O(n)
- **bignum_fromstr:** O(n)
- **bignum_frombignum:** O(n)
- **bignum_tostr:** O(n)
- **bignum_tobytes:** O(n)
- **bignum_print:** O(1)
- **bignum_deinit:** O(1)
- **bignum_add:** O(n)
- **bignum_sub:** O(n)
- **bignum_imultiply:** O(n)
- **bignum_idivide:** O(n)
- **bignum_imod:** O(n)
- **bignum_imodinv:** O(n)
- **bignum_less:** O(1)

n = le nombre de caracteres de message.

Donc:

- La complexité temporelle totale d'algorithme de **cryptage** est de **O(n^2)**.
- La complexité spatiale totale d'algorithme de **cryptage** est de **O(n)**.
- La complexité temporelle totale d'algorithme de **décryptage** est de **O(n^3)**.
- La complexité spatiale totale d'algorithme de **décryptage** est de **O(n)**.
- La complexité temporelle totale d'algorithme de **génération de clé** est de **O(n^3)**.
- La complexité spatiale totale d'algorithme de **génération de clé** est de **O(n)**.

Complexité pratique

- **bignum_fromint:** O(1)
- **bignum_frombytes:** O(n)
- **bignum_fromstr:** O(n)
- **bignum_frombignum:** O(n)
- **bignum_tostr:** O(n)

- **bignum_tobytes:** O(n)
- **bignum_print:** O(1)
- **bignum_deinit:** O(1)
- **bignum_add:** O(n)
- **bignum_sub:** O(n)
- **bignum_imultiply:** O(n^2)
- **bignum_idivide:** O(n^2)
- **bignum_imod:** O(n^2)
- **bignum_imodinv:** O(n^3)
- **bignum_less:** O(1)

Donc:

- La complexité pratique totale d'algorithme de **cryptage** est de **O(n^2)**.
- La complexité pratique totale d'algorithme de **décryptage** est de **O(n^3)**.
- La complexité pratique totale d'algorithme de **génération de clé** est de **O(n^3)**.

Complexité pratique

Je n'ai pas bien compris la question des graphiques donc c'est ce qui me vient à l'esprit comme réponse à cette question :

- **Génération de clé:** T(n) = a*n^3 + b*n^2 + c*n + d
- **Cryptage:** T(n) = a*n^2 + b*n + c
- **Décryptage:** T(n) = a*n^3 + b*n^2 + c*n + d