TP 2 : Cryptographie Classique

Exercice 1 :

Ecrire une fonction inv=invmod(x,n) qui calcule l'inverse modulaire d'un nombre entier x dans Zn.
Calculer à l'aide de cette fonction l'inverse dans Z_{26} des nombres suivants : 3, 5 , 9
Fonctions Matlab utiles :
d=gcd(a,n) : retourne le pgdc entre a et n, si d=1 alors a et n sont des nombres premiers.
mod(a,n) : retourne le résultat de *a modulo n*.

Exercice 2 :

L'algorithme de chiffrement Affine est basé sur la fonction affine : Y=ax+b mod N

a) Ecrire la fonction de chiffrement c=affine_E(m,k), chiffrer avec cette fonction le message suivant : « chiffrement-affine» avec la clé (3,5)
b) En utilisant la fonction inv=invmod(x,n) de l'exercice 1, écrire la fonction de déchiffrement m=affine_D(c,k), en utilisant cette fonction, déchiffrez avec la clé (7,11) le message suivant :
   `6O+Dr+!vDK/3+kvrA3+Dr+!yDd3=+%\36=+(D3+d\Dr+,36+,3DY+36=+kvdv,3`
c) Soit le message chiffré avec la clé (9,3) « 3/#A%iwNrw{S#Sr{#%E#.w3/wE#JS\/%{ », déchiffrer ce message à l'aide de la fonction affine_E(m,k)

Exercice 3 :

Donnez le code MATLAB qui correspond aux différentes fonctions de l'algorithme RSA, en l'occurrence :

function Result = ModularExponentiation(Base, Exponent, Modulus)
function [Modulus, PublicKey, PrivateKey] = GenerateKeyPair
function Ciphertext = Encrypt(Modulus, PublicKey, Message)
function Message = Decrypt(Modulus, PrivateExponent, Ciphertext)
function Signature = Sign(Modulus, PrivateExponent, Message)
function IsVerified = Verify(Modulus, PublicExponent, Message, Signature)

Fonctions Matlab utiles :
OUT = RANDSEED(RANDSEED, M, N, RMIN, RMAX) génère une matrice MxN de nombres premiers dans l'intervalle [RMIN, RMAX].