Master CSI 1 2015-2016

Courbes elliptiques — N1MA8W04

Responsables : G. Castagnos, D. Robert

ECDSA

Le NIST spécifie plusieurs courbes elliptiques pour des utilisations cryptographiques. Voir par exemple `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`, appendice D.1. On va utiliser la courbe dite P – 256 définie sur $\mathbf{F}_p$ avec p un nombre premier de 256 bits. Cette courbe a pour équation $y^2 = x^3 - 3x + b$ avec

$$p = 115792089210356248762697446949407573530086143415290314195533631308867097853951$$

$$b = 41058363725152142129326129780047268409114441015993725554835256314039467401291$$

1 Définir cette courbe avec Pari/GP. Le NIST annonce que le groupe des points sur $\mathbf{F}_p$ de cette courbe est cyclique d'ordre premier

$$n = 115792089210356248762697446949407573529996955224135760342422259061068512044369$$

Retrouver ce nombre avec Pari/GP.

2 Le NIST donne comme générateur le point $P = (x, y)$ avec

$$x = 48439561293906451759052585252797914202762949526041747995844080717082404635286$$

et

$$y = 36134250956749795798585127919587881956611106672985015071877198253568414405109$$

Vérifier que P est bien un point de la courbe. Implanter l'algorithme d'exponentiation *double and add*. Calculer nP.

3 Simuler un échange de clef par Diffie-Hellman.

4 Programmer l'algorithme de signature ECDSA. Pour la fonction de hachage on utilisera la fonction suivante qui fait appel à SHA256 :

```
hash(x)=
{
return(extern("python -c \"import hashlib;print(int(hashlib.sha256('"x"').hexdigest(),16))\""))
}
```

5 Programmer une fonction prenant en entrée deux entiers k et ℓ avec $\ell \leqslant k$ et ressortant E, p, q, P tels que E soit une courbe elliptique aléatoire sur $\mathbf{F}_p$ avec p premier de k bits, q soit un nombre premier de ℓ bits divisant l'ordre de la courbe $E(\mathbf{F}_p)$ et P un point de $E(\mathbf{F}_p)$ d'ordre q. Adapter la méthode ρ de Pollard pour une telle courbe.