Université de Limoges **Année universitaire 2015-2016**
Faculté des Sciences et Techniques

Master 2 Math - CRYPTIS

Examen TP - Cryptographie à clée secrète

- Durée 3h.
- Manuscrits et documents autorisés. Les documents du voisin ne sont pas des documents autorisés.
- Pour chaque programme, on rédigera des explications et on joindra le fichier source magma associé
- Les documents électroniques sont à envoyer par mail à `duong-hieu.phan@unilim.fr` avant la fin de l'épreuve

Dual EC_DRBG - Dual Points on an Elliptic Curve for Deterministic Random Bit Generation. L'objectif est d'étudier le générateur pseudo-aléatoire $DualEC\_DRBG$ introduit par NIST. Il est récemment détecté que Dual EC_DRBG contient une “backdoor”.
En général, le Dual EC_DRBG est décrit comme suit :

- Considérons une courbe elliptique (E) :$y^2 = x^3 + ax + b$ sur un corps fini $\mathbb{F}_p$ (p est premier).
- Considérons G, un sous groupe du groupe de points de E, d'ordre n premier
- Soit $P, Q \in G$
- Soit s un grain aléatoire ($s < n$)
- Mettons $s_0 = s$
- Pour chaque $i > 0$ on définit :
  - r_i est $x-$coordonnée de $s_{i-1}P$
  - s_i est $x-$coordonnée de r_iP
  - t_i est $x-$coordonnée de r_iQ
- La suite pseudo-aléatoire est définie comme $t_1, t_2, t_3, ...$
  (dans la version complète de NIST, on enlève les 16 premiers bits de chaque t_i. Ici, pour simplifier, on prend tous les bits de t_i).

Programme 1 [4 points]. Ecrire un premier programme Magma (avec le nom VotreNom-Prog1) pour simuler la version 256 bits avec les paramètres suivants :

- $p := 115792089210356248762697446949407573530086143415290314195533631308867097853951$;
- $a := -3; b := 0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b$;
- Calculer n qui est l'ordre de E
- Vérifier que l'ordre de E est premier. Alors, on peut définir $G = E$
- Choisir aléatoirement P dans E
- Choisir aléatoirement $1 \leq k \leq n$
- Définir $Q := kP$
- Afficher P et Q

Programme 2 [5 points]. Ecrire un deuxième programme Magma (avec le nom VotreNom-Prog2) pour simuler la génération des bits pseudo-aléatoires :

— En entrée : la courbe E et P, Q du Programme 1

— Générer aléatoirement un grain $1 \leq s \leq n$

— Générer la suite $t_1, t_2, t_3, ..., t_{100}$

Programme 3 [11 points]. Ecrire un troisième programme Magma (avec le nom VotreNom-Prog3) pour simuler la génération des bits pseudo-aléatoires avec k tel que $Q = kP$, sans avoir besoin de connaître le grain s :

— En entrée : la courbe E et P, k du Programme 1 ; deux valeurs t_1, t_2 retournées du Programme 2

— Calculer $t_3, t_4, ..., t_{110}$

Quelques instructions de Magma

— $ElementToSequence(P)$: pour un point $P(x : y : 1)$, ça donne une séquence $[x, y, 1]$

— Créer une suite : $t := AssociativeArray()$

— Faire attention au type des variables. Example : rP est correct pour un entier r mais pas correct pour un $r \in \mathbb{F}_p$. Pour changer de type, on peut utiliser par exemple $r := IntegerRing()!r$;

— $Points(E, x)$ donne tous les points sur E avec le coordonnée x.

— $P := elt < E|x, y, z >$ retourne le point $P(x : y : z)$ sur la courbe E ou retourne une erreur si $(x : y : z)$ n'est pas sur E. Si z n'est pas précisé alors $P := elt < E|x, y >$ retourne le point $P(x : y : 1)$

Remarque finale : Si P et Q sont tiré aléatoirement et indépendemment alors Dual EC_DRBG n'est pas vulnérable. Cependant, si on génère $Q = kP$ à partir d'un k alors k est un "backdoor" pour casser Dual EC_DRBG.