TP3 - Correction

9 Octobre 2020

Exercice 1 : Manipulation de points sur une courbe elliptique

1. L'implémentation est disponible dans le fichier `correction.sage`.
   (a) -
   (b) -
   (c) On remarque que l'ordre du point divise celui du groupe. Ici, le groupe est d'ordre 24, donc l'ordre du point généré peut être 1, 2, 4, 6, 12 ou 24 (ici il n'y a pas de point d'ordre 24).
   (d) Ici, l'approche la plus simple est de calculer kP en incrémentant k jusqu'à tomber sur l'élément neutre (noté $(0:1:0)$ dans `sage`). Le k correspondant est l'ordre du point P.
   On sait toutefois que l'ordre de P divise celui du groupe. On peut donc se contenter de tester les k divisant 24.
   (e) La coordonnée est x correspond à celle d'un point si elle permet de satisfaire l'équation de la courbe sur $\mathbb{Z}/29\mathbb{Z}$ pour un $y \in \mathbb{Z}/29\mathbb{Z}$. Il faut donc commencer par calculer $x^3 - 3x + 5$ (toujours sur $\mathbb{Z}/29\mathbb{Z}$) et vérifier si le résultat correspond à un carré modulo 29.
   Attention, si vous utilisez la méthode `is_square` sur un entier "classqiue", le résultat ne sera pas forcement bon. Il faut vérifier si il s'agit d'un carré **modulo 29**. Un autre méthde pour déterminer s'il s'agit d'un carré est `legendre_symbol` ou `jacobi_symbol` qui retournent 1 si il s'agit d'un carré.
   (f) On peut simplement parcourir les coordonnées en x de 0 à 29,et vérifer s'il existe un point correspondant. Si c'est la cas, la coordonnée en x peut correspondre à deux points : (x, y) et $(x, -y)$. N'oubliez pas le point infini.
2. Considérons maintenant la courbe définie par $y^2 = x^3 - 3x + 6$ sur les entiers modulo

$$p = 5164669856444950218363050899868468345 3$$

   Cette courbe comporte $\ell = 5164669856444950218892505989770713344 1$ points (elle est donc d'ordre ℓ).
   (a) L'ordre du point divise l'ordre de la courbe, et $\ell = 743 \times 69511034407065278854542476309161687$, d'où $ord(P) \in \{1, 743, 69511034407065278854542476309161687, \ell\}$.
   (b) P est d'ordre 743.
   (c) L'ordre du point est petit. Il ne permet donc que de générer peu de points. Dans ce contexte, il est possible de résoudre le logarithme discret, rendant les applications cryptographiques non sécurisées.
   (d) Les meilleurs algorithmes connus pour résoudre ECDLP tourne en $\mathcal{O}(\sqrt{n})$, où n représente l'ordre de l'élément. Ici, $n = \ell \approx 2^{125}$, offrant seulement $\sqrt{2^{125}} = 2^{64,5}$ bits de sécurité.
   La sécurité attentu aujourd'hui doit être supérieur à 2^{128}, c'est pourquoi les courbes couramment utilisées ont un ordre approchant de 2^{256} ou plus.

Exercice 2 : ECDSA

1. Multiples sources disponibles en ligne. Gardez en tête que intégrité $\neq$ authenticité.

   L'idée est que ECDSA est une fonction de signature, permettant d'authentifier un message. C'est à dire qu'un utilisateur ayant une clé (d, P), avec d privé et $P = d \times G$ publique, peut signer un message m a l'aide de sa clé **privée**. La signature consiste en deux entiers (r, s), considérés publiques.

   Tout utilisateur en possession du message m, de la clé publique P et de la signature (r, s) est en mesure de vérifier si la signature est valide.

2. Cf. correction.sage

3. Le nonce est effictivement sensible. Il doit être secret, et différent pour chaque signature effectuée avec la même clé. Toute fuite d'information (même quelques bits) sur le nonce compromet la confidentialité de la clé privée.

Exercice 3 : Attaques sur ECDSA (Bonus)

Définissons les notations suivantes :
- k est le nonce
- G est le point générateur, publique, utilisé par tout les utilisateur de la courbe $P256$. G est d'ordre n et $P256$ est définie sur $\mathbb{Z}/p\mathbb{Z}$.
- La bi-clé d'un utilisateur A est notée (d_A, P_A) avec $P_A = d_A \times G$.
- Si $P = (x, y)$ est un point sur la courbe, P_x désigne sa coordonnée en x.

1. Supposez qu'un utilisateur signe un message et le transmette avec le nonce utilisé durant la signature.
   (a) Un attaquant peut retrouver la clé privée : Sachant que $s = k^{-1}(h(m) + d \times r) \bmod n$, la seule inconnue est d, la clé privée. On peut simplement inverser les opérations une par une pour arriver au résultat (toute les opération si dessous sont effectuées modulo n) :

   $$\begin{aligned} s &= k^{-1}(h(m) + d \times r) \\ s \times k &= h(m) + d \times r \\ s \times k - h(m) &= d \times r \\ (s \times k - h(m))r^{-1} &= d \end{aligned}$$

   (b) Cf correction.sage

2. Cette fois, l'utilisateur a gardé le nonce secret, mais l'a utilisé pour signer plusieurs messages différents (deux en l'occurrence). Ce problème est connu et a causé de nombreux problèmes dans des produits déployés, notamment chez la PS3 [1, 2], compromettant entièrement la console.
   (a) La valeur de r sera la même car $r = k \times P$. Ici, les message m2_1 et m2_3 sont concernés.
   (b) On sait que $r_1 = r_2 = r$. $s_1 = k^{-1}(h(m_1) + d \times r)$ et $s_2 = k^{-1}(h(m_2) + d \times r)$. En soustrayant les deux valeurs de s_1 ety s_2, on peut retrouver la valeur du nonce. À partir de là, la même attaque s'applique.

   Les opérations ci-dessous sont toutes effectuées modulo n.

   $$\begin{aligned} s_1 - s_2 &= k^{-1}(h(m_1) + d \times r) - k^{-1}(h(m_2) + d \times r) \\ &= k^{-1}(h(m_1) - h(m_2)) \\ \Rightarrow k &= (s_1 - s_2)^{-1} \times (h(m_1) - h(m_2)) \end{aligned}$$

   (c) Cf correction.sage

1. https://media.ccc.de/v/27c3-4087-en-console_hacking_2010
2. https://www.bbc.com/news/technology-12116051