Algorithme d'exponentiation rapide

Pour chiffrer et déchiffrer des messages en appliquant l'algorithme RSA, nous avons besoin de calculer rapidement des puissances modulo n. Pour cela il existe une méthode beaucoup plus efficace que de calculer d'abord a^k puis de le réduire modulo n. (*Il faut garder à l'esprit que les entiers que l'on va manipuler ont des dizaines voire des centaines de chiffres.*)

Voyons la technique sur l'exemple de $5^{11} \pmod{14}$. L'idée est de seulement calculer $5, 5^2, 5^4, 5^8 \ldots$ et de réduire modulo n à chaque fois. Pour cela on remarque que $11 = 8+2+1$ donc

$$5^{11} = 5^8 \times 5^2 \times 5^1.$$

Calculons donc les $5^{2^i} \pmod{14}$:

$$\begin{aligned}
&5 \equiv 5 \pmod{14} \\
&5^2 \equiv 25 \equiv 11 \pmod{14} \\
&5^4 \equiv 5^2 \times 5^2 \equiv 11 \times 11 \equiv 121 \equiv 9 \pmod{14} \\
&5^8 \equiv 5^4 \times 5^4 \equiv 9 \times 9 \equiv 81 \equiv 11 \pmod{14}
\end{aligned}$$

à chaque étape est effectuée une multiplication modulaire. Conséquence :

$$5^{11} \equiv 5^8 \times 5^2 \times 5^1 \equiv 11 \times 11 \times 5 \equiv 11 \times 55 \equiv 11 \times 13 \equiv 143 \equiv 3 \pmod{14}.$$

Nous obtenons donc un calcul de $5^{11} \pmod{14}$ en 5 opérations au lieu de 10 si on avait fait $5 \times 5 \times 5 \cdots$.

Voici une formulation générale de la méthode. On écrit le développement de l'exposant k en base 2 : $(k_\ell, \ldots, k_2, k_1, k_0)$ avec $k_i \in \{0,1\}$ de sorte que

$$k = \sum_{i=0}^{\ell} k_i 2^i.$$

On obtient alors

$$x^k = x^{\sum_{i=0}^{\ell} k_i 2^i} = \prod_{i=0}^{\ell} (x^{2^i})^{k_i}.$$

Par exemple 11 en base 2 s'écrit (1,0,1,1), donc, comme on l'a vu :

$$5^{11} = (5^{2^3})^1 \times (5^{2^2})^0 \times (5^{2^1})^1 \times (5^{2^0})^1.$$

Voici un autre exemple : calculons $17^{154} \pmod{100}$. Tout d'abord on décompose l'exposant $k = 154$ en base 2 : $154 = 128 + 16 + 8 + 2 = 2^7 + 2^4 + 2^3 + 2^1$, il s'écrit donc en base 2 : (1,0,0,1,1,0,1,0).

Ensuite on calcule $17, 17^2, 17^4, 17^8, \ldots, 17^{128}$ modulo 100.

$$\begin{aligned}
&17 \equiv 17 \pmod{100} \\
&17^2 \equiv 17 \times 17 \equiv 289 \equiv 89 \pmod{100} \\
&17^4 \equiv 17^2 \times 17^2 \equiv 89 \times 89 \equiv 7921 \equiv 21 \pmod{100} \\
&17^8 \equiv 17^4 \times 17^4 \equiv 21 \times 21 \equiv 441 \equiv 41 \pmod{100} \\
&17^{16} \equiv 17^8 \times 17^8 \equiv 41 \times 41 \equiv 1681 \equiv 81 \pmod{100} \\
&17^{32} \equiv 17^{16} \times 17^{16} \equiv 81 \times 81 \equiv 6561 \equiv 61 \pmod{100} \\
&17^{64} \equiv 17^{32} \times 17^{32} \equiv 61 \times 61 \equiv 3721 \equiv 21 \pmod{100} \\
&17^{128} \equiv 17^{64} \times 17^{64} \equiv 21 \times 21 \equiv 441 \equiv 41 \pmod{100}
\end{aligned}$$

Il ne reste qu'à rassembler :

$$17^{154} \equiv 17^{128}\times17^{16}\times17^{8}\times17^{2} \equiv 41\times81\times41\times89 \equiv 3321\times3649 \equiv 21\times49 \equiv 1029 \equiv 29 \pmod{100}$$

À partir des questions précédente écrire une fonction Python `p=powermod(a,e,n)` calculant $a^e \mod n$, basée sur le principe d'exponentiation rapide (« Square & Multiply »)
.

En fait Python sait faire l'exponentiation rapide : `pow(x,e,n)` *pour le calcul de a^e modulo n, il faut donc éviter* `(x ** e) \% n` *qui n'est pas adapté.*