Attaque RSA

R3.09 Cryptographie
T. Godin, L. Naert, T. Ferragut
IUT de Vannes, Département Informatique

Rappels

Préparation du protocole

Bob choisit [1] $\mathbf{p}, \mathbf{q}$ deux (grands) nombres premiers distincts.

Il peut alors calculer (facilement) $\varphi(\mathbf{n}) = (\mathbf{p}-\mathbf{1})(\mathbf{q}-\mathbf{1})$.

Il choisit ensuite $\mathbf{d}, \mathbf{e}$ tels que $\mathbf{d}.\mathbf{e} \equiv \mathbf{1} \mod (\varphi(\mathbf{n}))$.

On a donc les clefs

- publique : $(\mathbf{e}, \mathbf{n})$
- privée : $(\mathbf{p}, \mathbf{q}, \mathbf{d})$

On remarquera que les connaissances privées de **Bob** permettent le chiffrement et le déchiffrement efficaces, tandis que les connaissances publiques ne permettent que le chiffrement.

Échange d'un message

Dans un protocole *RSA*, si **Alice** veut envoyer un message $\mathbf{m}$ à **Bob**, elle utilise la *clef publique* $(\mathbf{n}, \mathbf{e})$ pour envoyer le message chiffré $\mathbf{c} \equiv \mathbf{m}^{\mathbf{e}} \ \mathbf{mod}(\mathbf{n})$.

Bob reçoit le message $\mathbf{c}$ et le *déchiffre* en utilisant $\mathbf{m} \equiv \mathbf{c}^{\mathbf{e}} \mod (\mathbf{n})$.

En effet, d'après le théorème d'Euler [2], comme $\mathbf{e}.\mathbf{d} \equiv \mathbf{1} \mod (\varphi(\mathbf{n}))$, on a :

$$\mathbf{c}^{\mathbf{e}} \equiv \left(\mathbf{m}^{\mathbf{d}}\right)^{\mathbf{e}} \equiv \mathbf{m}^{\mathbf{de}} \equiv \mathbf{m}^{\mathbf{k}\varphi(\mathbf{n})+\mathbf{1}} \equiv \mathbf{1}.\mathbf{m} \equiv \mathbf{m} \mod (\mathbf{n})$$

Bob obtient donc bien le message $\mathbf{m}$ qu'**Alice** souhaitait lui envoyer.

---

1. Dans tout le texte, les informations publiques (ou non sécurisées, ce qui est presque la même chose en cryptographie) seront écrites en vert, les informations privées seront en bleu et les informations provenant d'une attaque seront en rouge. Les calculs (*a priori* effectués par un ordinateur) seront `notés sur fond gris`.
2. **Théorème d'Euler** : Si a est un entier premier avec n, alors $a^{\varphi(n)} \equiv 1 \mod (n)$. Dans notre cas, comme $n = p.q$, cela fonctionne même si $a < n$ n'est pas premier avec n.

Un exemple d'attaque

On se place dans une situation où un adversaire, **Oscar**, espionne **Alice** et **Bob**.

Oscar dispose des informations visibles sur le réseau, en particulier de la clef publique, et il voit passer les messages chiffrés. Il possède également un (super)ordinateur lui permettant d'essayer d'effectuer des factorisations par brute-force.

Ici, supposons que la clef publique soit $(\mathbf{e}, \mathbf{n}) = (\mathbf{3}, \mathbf{55})$ et qu'**Alice** envoie le message chiffré $\mathbf{c} = \mathbf{12}$. **Oscar** va essayer de décrypter le message.

- **étape 1 :** **Oscar** essaie de factoriser $\mathbf{n} = \mathbf{55}$. À l'aide de son supercalculateur, il trouve $\mathbf{p} = \mathbf{5}, \mathbf{q} = \mathbf{11}$
- **étape 2 :** **Oscar** en déduit la fonction indicatrice $\varphi(\mathbf{55}) = (\mathbf{p} - \mathbf{1})(\mathbf{q} - \mathbf{1}) = (\mathbf{5} - \mathbf{1})(\mathbf{11} - \mathbf{1}) = \mathbf{40}$
- **étape 3 :** À l'aide, par exemple, d'une fonction `coefBezout`, il calcule l'inverse de $\mathbf{e}$ modulo $\varphi(\mathbf{n})$: $\mathbf{d} = \mathbf{e}^{-\mathbf{1}} \mod (\varphi(\mathbf{n}))$. Ici $\mathbf{d} = \mathbf{27}$
- **étape 4 :** Il ne reste plus à **Oscar** qu'à calculer $\mathbf{c}^{\mathbf{d}} \mod (\mathbf{n})$ (cela peut se faire efficacement avec une fonction `exponentitionRapideMod`) pour trouver $\mathbf{m} \equiv \mathbf{12}^{\mathbf{27}} \equiv \mathbf{23} \mod (\mathbf{n})$.

Le message envoyé par **Alice** était donc $\mathbf{m} = \mathbf{23}$