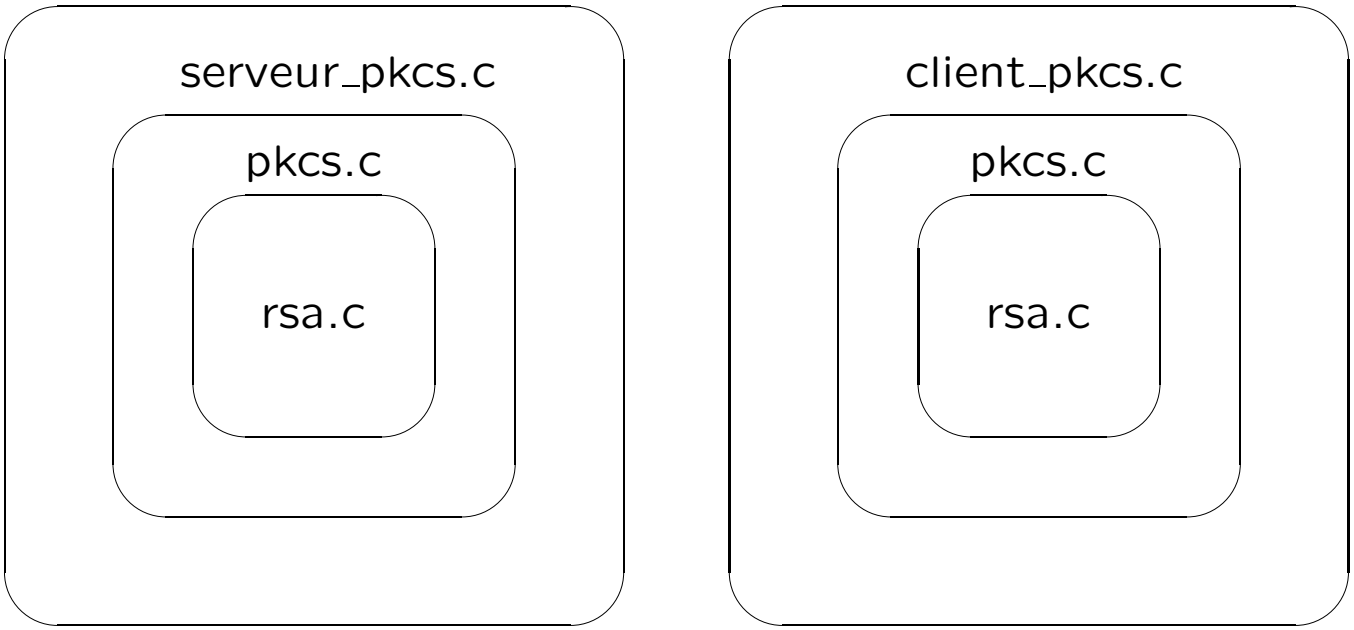
L'implémentation (4)



L'implémentation (2)

Version normale du client :

- Implémente l'attaque telle qu'elle est décrite dans l'algorithme
- Nécessite beaucoup de temps pour être éxecutée

Version Locale :

- Nécessite la clé privée d passée en paramètre
- d n'est pas utilisé pour déchiffrer m !
- d permet de simuler la vérification faite par le serveur
- on forme $m_0 \cdot s$ plutot que $c_0 \cdot s^e \Rightarrow$ on économise un calcul de puissance

L'implémentation (1)

- Le programme qui implémente l'attaque a été écrit en C.
- Utilisation librairie gmp de GNU pour calculs sur grands nombres
- Application client/serveur. Le client attaque le serveur comme décrit dans l'algorithme
- 2 versions du client, une normale (3 jours pour dechiffrer un message), une rapide (15 minutes). Bien entendu la deuxième triche . . .

Mais c'est peine perdue... (2)

Dans le meilleur des cas, la solution proposée au paragraphe précédent permettra à l'algorithme de continuer. Les deux entiers a et b ne pouront que se rapprocher. La quantité $\frac{as-3B+1}{n}$ va croitre, la quantité $\frac{bs-2B}{n}$ va décroitre. Comme l'inégalité

$$\frac{as-3B+1}{n} < \frac{bs-2B}{n}$$

est toujours vérifiée, ces deux quantités seront de plus en plus proches l'une de l'autre.
Il sera de plus en plus difficile de trouver un entier s à la fois acceptable au pas 2 et tel que :

$$\left\lceil \frac{as-3B+1}{n} \right\rceil \leq \left\lfloor \frac{bs-2B}{n} \right\rfloor$$

Mais c'est peine perdue... (1)

En effet si le problème se présente cela signifie que :

$$\left\lceil \frac{as - 3B + 1}{n} \right\rceil > \left\lfloor \frac{bs - 2B}{n} \right\rfloor$$

Or, on a toujours :

$$\frac{as - 3B + 1}{n} < \frac{bs - 2B}{n}$$

Cela signifie que :

$$\frac{as - 3B + 1}{n} \approx \frac{bs - 2B}{n}$$

Solution sans réinitialisation (4)

Si l'on remplace α et η' par leur valeur cela donne :

$$\frac{b}{n}\eta > \left\lceil \frac{bs-2B}{n} \right\rceil - \frac{bs-2B}{n} \Rightarrow \eta > \frac{n}{b}\left(\left\lceil \frac{bs-2B}{n} \right\rceil - \frac{bs-2B}{n} \right)$$

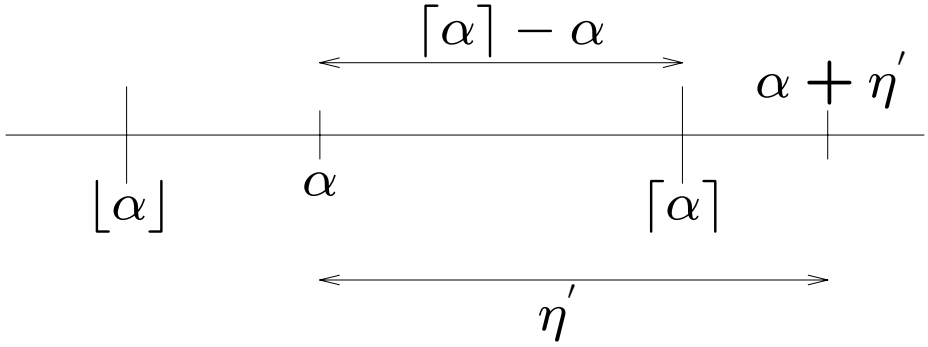
Et comme η est entier :

$$\eta \geq \left\lceil \frac{n}{b}\left(\left\lceil \frac{bs-2B}{n} \right\rceil - \frac{bs-2B}{n} \right) \right\rceil$$

On voit que si une valeur s trouvée au deuxième pas de l'algorithme pose problème, on cherche une nouvelle valeur s' telle que :

$$s' \geq s + \left\lceil \frac{n}{b}\left(\left\lceil \frac{bs-2B}{n} \right\rceil - \frac{bs-2B}{n} \right) \right\rceil$$

Solution sans réinitialisation (3)



Notons que :

$$\left\lfloor \alpha + \eta' \right\rfloor > \lfloor\alpha\rfloor \Leftrightarrow \eta' > \lceil\alpha\rceil - \alpha$$

Solution sans réinitialisation (2)

Notons $\eta = s' - s$. L'inégalité précédente donne :

$$\left\lfloor \frac{b(s+\eta) - 2B}{n} \right\rfloor > \left\lfloor \frac{bs - 2B}{n} \right\rfloor \Rightarrow \left\lfloor \frac{bs - 2B}{n} + \frac{b}{n}\eta \right\rfloor > \left\lfloor \frac{bs - 2B}{n} \right\rfloor$$

Si on note $\alpha = \frac{bs-2B}{n}$ et $\eta' = \frac{b}{n}\eta$, la dernière inégalité donne :

$$\left\lfloor \alpha + \eta' \right\rfloor > \lfloor \alpha \rfloor$$

Solution sans réinitialisation (1)

Le problème survient au pas 3 quand pour l'un des intervalles $[a, b]$ de M_i :

$$\left\lceil \frac{as - 3B + 1}{n} \right\rceil > \left\lfloor \frac{bs - 2B}{n} \right\rfloor$$

Les deux termes de cette inégalité sont croissant en fonction de s. Elle ne peut donc pas s'inverser tant que le terme de droite n'a pas augmenté. La nouvelle valeur s' de s devra donc vérifier l'inégalité suivante :

$$\left\lfloor \frac{bs' - 2B}{n} \right\rfloor > \left\lfloor \frac{bs - 2B}{n} \right\rfloor$$

Solutions

Deux solutions sont envisageables si le problème se pose :

- 1ère Solution :
  Réinitialiser l'algorithme

- 2ème Solution :
  Ignorer la valeur de s trouvée, en checher une autre que l'on appelle s'. On cherche la distance minimale entre s' et s pour que le problème puisse disparaitre.

Problème rencontré au niveau du 3ème pas (2)

La première inéquation est toujours vérifiée, en effet :

$$\begin{cases} B \geq 1 \\ a < b \end{cases} \Rightarrow as - 3B + 1 \leq bs - 2B \Rightarrow \frac{as - 3B + 1}{n} \leq \frac{bs - 2B}{n}$$

En revanche avec $\frac{as-3B+1}{n} = 1.4$ et $\frac{bs-2B}{n} = 1.6$. Dans ce cas on chercherait un r tel que :

$$2 \leq r \leq 1 \text{ car } \left\lceil \frac{as - 3B + 1}{n} \right\rceil = 2 \text{ et } \left\lfloor \frac{bs - 2B}{n} \right\rfloor = 1$$

Ce qui est impossible.

Il arrive donc que l'algorithme se retrouve bloqué.

Problème rencontré au niveau du 3ème pas (1)

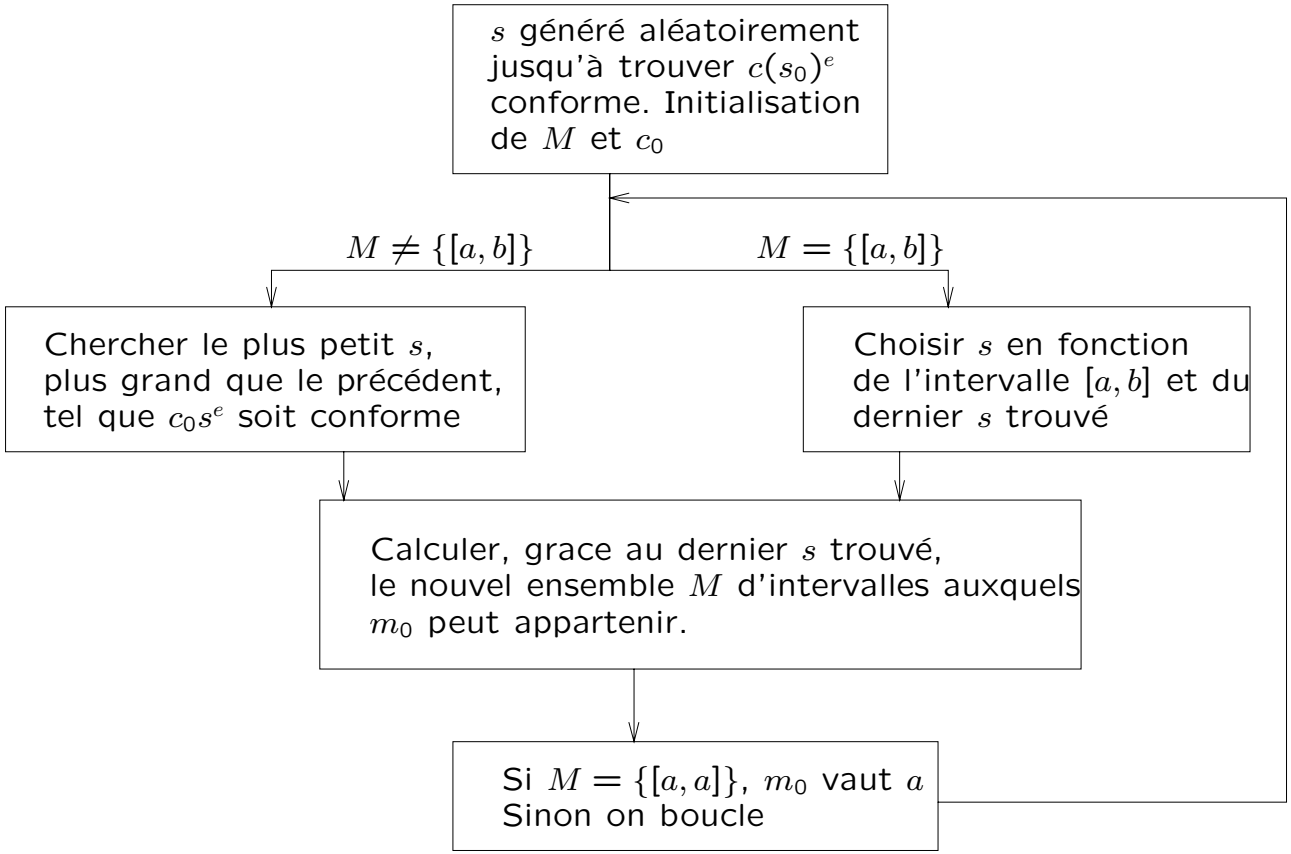
Dans le rapport original de Daniel Bleichenbacher, il faut chercher à un certain moment de l'algorithme un entier r tel que :

$$\frac{as - 3B + 1}{n} \leq r \leq \frac{bs - 2B}{n} \Rightarrow \frac{as - 3B + 1}{n} \leq \frac{bs - 2B}{n}$$

alors que dans le rapport de mon projet cette condition devient :

$$\left\lceil \frac{as - 3B + 1}{n} \right\rceil \leq r \leq \left\lfloor \frac{bs - 2B}{n} \right\rfloor \Rightarrow \left\lceil \frac{as - 3B + 1}{n} \right\rceil \leq \left\lfloor \frac{bs - 2B}{n} \right\rfloor$$

L'attaque - L'algorithme résumé



L'attaque - L'idée centrale (2)

On en déduit l'intervalle auquel appartient le nombre $m \cdot s$. On note $B = 2^{8(k-2)}$.
La plus petite valeur possible sera :

$$00||02||(k-2) \text{ bytes } \texttt{0x00} = 2 \cdot 2^{8(k-2)} = 2B$$

La plus grande valeur possible sera :

$$00||02||(k-2) \text{ bytes } \texttt{0xFF} = 3 \cdot 2^{8(k-2)} - 1 = 3B - 1$$

On a donc :

$$2B \leq ms \pmod{n} < 3B$$

En appliquant le même procédé un certain nombre de fois, on réduit l'intervalle jusqu'à connaître la valeur exacte de $m \cdot s$ et donc de m.

L'attaque - L'idée centrale (1)

Soit un texte chiffré c que l'on cherche à déchiffrer. On choisi un entier s et on calcule : $c' = c \cdot s^e \pmod{n}$.
On envoie c' à l'Oracle. Si ce dernier n'envoie pas de message d'erreur on sait que les deux premiers bytes du message en clair qui correspond à c' sont `0x00` et `0x02`. Si on déchiffre c' on obtient :

$$m' \equiv (c')^d \equiv (cs^e)^d \equiv c^d s^{ed} \equiv m \cdot s \pmod{n}$$

Description de PKCS#1 (2)

Un bloc EB de k bytes :

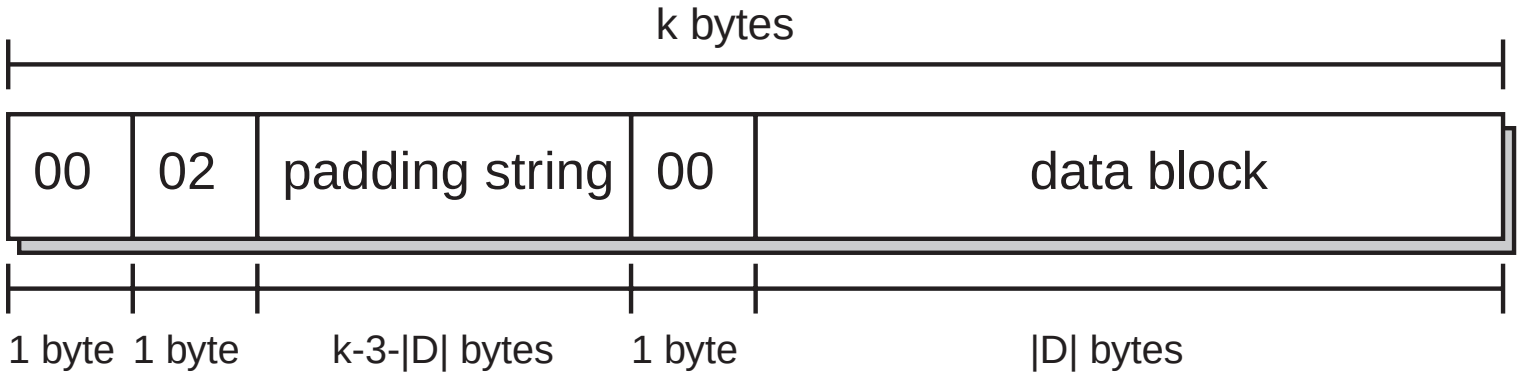
$$EB = EB_1||\ldots||EB_k$$

est conforme à PKCS#1 s'il est de la forme du bloc précédemment décrit. En particulier, EB doit satisfaire les conditions suivantes :

- $EB_1 = \texttt{0x00}$
- $EB_2 = \texttt{0x02}$
- EB_3 jusqu'à EB_{10} sont différents de `0x00`
- Au moins un des bytes EB_{11} à EB_k est `0x00`. Il permet de repérer le début du message.

Si le serveur PKCS#1 reçoit un message non conforme, il en informe le client. On applera ce serveur un Oracle.

Description de PKCS#1 (1)



On note k la longeur du modulus de RSA. Un bloc à chiffrer a aussi une longueur k. Le Padding String servira à obtenir un bloc de k bytes, quelle que soit la longueur du bloc de données. La longueur du Padding String PS doit être d'au moins 8 bytes, $|D|$ ne doit donc pas dépasser $k - 11$ bytes.

L'algorithme RSA

RSA est un algorithme de chiffrement à clé publique.

- p et q, nombres premiers de 512 bits
- Le modulus n est le produit des deux : $n = p \cdot q$
- La clé publique e est un nombre premier avec $\varphi(n) = (p-1)(q-1)$
- La clé privée d vaut : $d = e^{-1} \bmod (p-1)(q-1)$

Le chiffrement d'un message m en un message c et son déchiffrement s'effectuent de la manière suivante :

- Chiffrement : $c = m^e \bmod n$
- Déchiffrement : $m = c^d \bmod n$

L'attaque à texte chiffré choisi de Daniel Bleichenbacher contre les protocoles basés sur PKCS#1

Thomas Baignères, thomas.baigneres@epfl.ch

Ecole Polytechnique Fédérale de Lausanne