Cryptographie des cartes bancaires, fonctionnement et casse du chiffrement RSA

Grégoire PELLETIER

2020-2021

Problématique

Le chiffrement RSA va t-il demeurer un moyen efficace de protéger nos cartes bancaires dans le futur ?

1. Fonctionnement du chiffrement RSA
2. Limites de cette méthode
3. Solutions pour améliorer le chiffrement RSA

Fonctionnement général du chiffrement RSA

Le chiffrement RSA est asymétrique. Il est donc composé d'une clé publique pour chiffrer des données confidentielles et d'une clé privée qui permet de déchiffrer ces dites données. La clé privée fait aussi office d'identification (seul l'expéditeur possède la clé privée qui sert alors de signature).

Fonctionnement général du chiffrement RSA

Pour considérer le chiffrement fiable, il est nécessaire qu'on ne puisse pas déduire la clé privée grâce à la clé publique avec les méthodes calculatoires disponibles au moment de l'échange.

Fonctionnement détaillé du chiffrement RSA : création des clés

- On choisit p et q premiers et distincts
- On calcule n=p*q.
- On calcule l'indicatrice d'Euler de n : $\phi(n) = (p-1)(q-1)$.
- On choisit e un entier naturel premier avec $\phi(n)$, strictement inférieur à $\phi(n)$, e est l'exposant de chiffrement.
- On calcule l'exposant de déchiffrement d tel que $e * d \equiv 1 \pmod{\phi(n)}$(grâce à l'algorithme d'Euclide étendu).

Fonctionnement détaillé du chiffrement RSA : création des clés

Le couple (n, e) est la clé publique du chiffrement, la clé privée est le nombre d.

Fonctionnement détaillé du chiffrement RSA : chiffrement du message

Soit M un entier naturel représantant un message, M<n :
$C \equiv M^e \pmod{n}$.
C représente le message chiffré, C<n.

Fonctionnement détaillé du chiffrement RSA : déchiffrement du message

Pour déchiffrer C, on utilise d, on retrouve le message M avec :
$M \equiv C^d \pmod{n}$.

Fonctionnement détaillé du chiffrement RSA : exemple

- $p = 5$ et $q = 17$
- $n = p \times q = 85$
- $\varphi(n) = (p-1) \times (q-1) = \varphi(n) = 64$

- $e = 5$ et on a bien $\text{pgcd}(e, \varphi(n)) = \text{pgcd}(5, 64) = 1$
-
  - $5 \times 13 + 64 \times (-1) = 1$
  - donc $5 \times 13 \equiv 1 \pmod{64}$
  - l'inverse de e modulo $\varphi(n)$ est $d = 13$

- $m = 10$, $n = 85$ et $e = 5$
- $x \equiv m^e \pmod{n} \equiv 10^5 \pmod{85}$
  - $10^2 = 100 \equiv 15 \pmod{85}$
  - $10^4 = (10^2)^2 \equiv 15^2 \equiv 225 \equiv 55 \pmod{85}$
  - $10^5 = 10^4 \times 10 \equiv 55 \times 10 \equiv 550 \equiv 40 \pmod{85}$
- Le message chiffré est donc $x = 40$

- $x = 40$, $d = 13$, $n = 85$ $\qquad 40^{13} \pmod{85}$
  - $40^2 = 1\,600 \equiv 70 \pmod{85}$
  - $40^4 = (40^2)^2 \equiv 70^2 \equiv 4\,900 \equiv 55 \pmod{85}$
  - $40^8 = (40^4)^2 \equiv 55^2 \equiv 3\,025 \equiv 50 \pmod{85}$
- $40^{13} \equiv 40^{8+4+1} \equiv 40^8 \times 40^4 \times 40 \equiv 50 \times 55 \times 40 \equiv 10 \pmod{85}$
- On retrouve bien le message $m = 10$ de Bruno

Fonctionnement détaillé du chiffrement RSA : algorithme python

Shells

Python

```
>>> decoupe('Fluctuat nec mergitur',3)
['Flu', 'ctu', 'at ', 'nec', ' me', 'rgi', 'tur']

>>> m=decoupe('Fluctuat nec mergitur',3)

>>> cle()
{'priv': (1022117, 101), 'pub': (1022117, 767597)}

>>> c=chiffre(1022117, 101, m)

>>> dechiffre(1022117, 767597, c)
'Fluctuat nec mergitur'
```

Limites de cette méthode : longueur de n

L'efficacité du chiffrement RSA repose sur l'hypothèse que, pour les algorithmes classiques, le temps que prend la factorisation du nombre n croît exponentiellement avec la longueur de n.
A l'origine les clés utilisées faisaient 320 bits de long, puis, pour des raisons de sécurité, cette longueur est passée à 1024 puis 2048 bits et désormais, dans certains cas, 4096 bits.

Limites de cette méthode : longueur de n

En 1998, Serge Humpich est parvenu à contourner deux types de sécurité, le code confidentiel et le chiffrement RSA.
Les clés utilisaient alors ne faisait que 320 bits de long. Serge Humpich a pu obtenir la factorisation de n grâce à un simple logiciel (cela faisait 10 ans que les nombres de 320 bits était factorisables en temps raisonnable).

Limites de cette méthode : longueur de n

Grâce à une "YesCard" (transmet une autorisation de transfert quelque soit le code secret tapé par son titulaire), il a pu effectuer les transactions qui étaient protégées uniquement par ces deux types de sécurité.

Limites de cette méthode : algorithme python simple

```
>>> facteurs(12345678901234567890)
[2, 3, 3, 5, 101, 3541, 3607, 3803, 27961]

>>>facteurs(608271779807659874 3)
[1536921011, 3957729613]
```

temps de factorisation de 12345678901234567890 : moins d'une seconde
temps de factorisation de 608271779807659874 3 : plusieurs minutes

Limites de cette méthode : logiciel utilisant le crible algébrique

Le crible algébrique est à ce jour le meilleur algorithme pour factoriser des entiers, particulièrement au delà de 320 bits.
Complexité estimée du crible algébrique :

$$\mathcal{O}\left(e^{1.92(\ln n)^{1/3}(\ln \ln n)^{2/3}}\right)$$

Limites de cette méthode : logiciel utilisant le crible algébrique

En utilisant le logiciel msieve, qui utilise le crible algébrique, on peut factoriser des entiers de 256 bits facilement, on retrouve ainsi p et q en 2 minutes environ.
Exemple réalisé en retrouvant :
(p=292727782972497489806532560966393292437,
q=336047202777652025357855832985297665233)

Limites de cette méthode : progrès algorithmique

Le progrès des algorithmes de factorisation est une première menace pour le chiffrement RSA. Le crible algébrique, créé au cours des années 1990, a ainsi permis d'établir de nombreux records sur la factorisation d'entier. Par exemple, en 2019, il a permis la factorisation d'un entier de forme générale de 795 bits.

Limites de cette méthode : Algorithme de Shor

L'algorithme de Shor est un algorithme quantique qui permet de factoriser un entier naturel avec une complexité asymptotique bien inférieure.
Cependant il nécessite un ordinateur quantique suffisamment grand.
Les ordinateurs quantiques permettent donc, théoriquement, de casser le RSA par la force brute aisément. Cependant, actuellement ces ordinateurs génèrent des erreurs aléatoires qui les rendent inefficaces pour implémenter l'algorithme de Shor.

Limites de cette méthode : Algorithme de Shor

- On choisit a<N.
- On calcule PGCD(a,N).
- Si PGCD(a, N) $\neq 1$, alors PGCD(a,N) est un diviseur non trivial de N, le problème est résolu.
- Sinon, on recherche une période r, la période de la fonction $f : x \mapsto a^x \bmod N$.
- Si r est impair, on recommence à l'étape 1.
- Si $a^{r/2} \equiv -1 \pmod N$, on recommence à l'étape 1.
- Les facteurs de N sont PGCD($a^{r/2} + 1$, N) et PGCD($a^{r/2} - 1$, N), ce qui résout le problème.

Limites de cette méthode : Algorithme de Shor

Les ordinateurs quantiques interviennent dans la recherche de la période r, ainsi il n'est pas possible d'utiliser cet algorithme pour des grandes valeurs de n avec un ordinateurs classique.
Les progrés de l'informatique quantique représente donc la plus grande menace pour le chiffrement RSA.

Solutions pour améliorer le chiffrement RSA

Augmenter la taille de n est indispensable pour assurer la fiabilité du chiffrement RSA (un entier de 1024 bits n'est plus considéré comme fiable).
L'attaque par force brute pour casser le chiffrement RSA avec des ordinateurs classiques semble irréalisable avec les clés de 2048 voir 4096 bits utilisées désormais.
L'évolution de ce nombre de bits en accord avec les progrès informatiques et algorithmiques assure la fiabilité du RSA.

Solutions pour améliorer le chiffrement RSA

La sécurité des cartes bancaires est assurée par la multiplications des protocoles de sécurité (comme l'authentification DES).
Cependant, le chiffrement RSA semble très vulnérable au évolution de l'informatique quantique sur le long terme et la création d'un ordinateur quantique suffisamment grand et fiable signifirait la fin de l'utilisation du RSA.

Bibliographies

- article de Jacques Patarin *La cryptographie des cartes bancaires*, Pour la science (2002)
- article de Thomas Genet *Le protocole cryptographique de paiement par carte bancaire*, Interstices (2008)
- présentation de Anca Nitulescu *Cryptosystème RSA*
- *Histoire de la carte à puce du point de vue d'un cryptologue* , Louis Guillou (2004)
- *EMV flaws and fixes : vulnerabilities in smart card payment systems*, Steven J. Murdoch (2007)