Factorisation d'entiers RSA

Quentin Deschamps

Automne 2020 *M2 SRI, ISFA*

Problème

- Dans un chiffrement RSA, l'entier N est connu et produit de deux nombres premiers p et q inconnus. Si p et q sont connus il est possible de déchiffrer.

Problème

- Dans un chiffrement RSA, l'entier N est connu et produit de deux nombres premiers p et q inconnus. Si p et q sont connus il est possible de déchiffrer.
- Position de l'attaquant : on va essayer de factoriser N.

Problème

- Dans un chiffrement RSA, l'entier N est connu et produit de deux nombres premiers p et q inconnus. Si p et q sont connus il est possible de déchiffrer.
- Position de l'attaquant : on va essayer de factoriser N.
- Rappel : Factoriser N est une façon d'attaquer RSA mais ça n'est pas la seule.

Problème

- Dans un chiffrement RSA, l'entier N est connu et produit de deux nombres premiers p et q inconnus. Si p et q sont connus il est possible de déchiffrer.
- Position de l'attaquant : on va essayer de factoriser N.
- Rappel : Factoriser N est une façon d'attaquer RSA mais ça n'est pas la seule.
- Les entiers RSA sont les plus difficiles à factoriser.

Meilleurs résultats actuels

Révrier 2020

RSA-250 = 21403246502407449612644230728393335630086147151447550177977549208814180234471401366433455190958046796109928518724709145876873962619215573630474547705208051190564931066876915900197594056934574522305893259766974716817380693648946998715784949759374979374641352894770715802787901901705773890848250147429434472081168596320245323446302386235987526683477087376619255856946397988533673337202759497815655622601060535511422794076034476755466678452098702384172921003708025744867329688187756571898625803693206271 1

Meilleurs résultats actuels

Février 2020

Le nombre RSA-250 a été factorisé le 28 février 2020. C'est le produit de deux nombres premiers de 125 chiffres chacun.
Ce résultat a été obtenu avec un algorithme spécifique appelé le crible algébrique, et un logiciel open-source (CADO-NFS) que les chercheurs du LORIA et leurs collègues développent depuis 2007, et qui comporte de l'ordre de 400 000 lignes de code. Pour établir ce nouveau record, il aurait fallu faire travailler un ordinateur pendant 2700 années ! À la place, ce sont environ 10000 ordinateurs qui ont calculé pendant quelques mois.

Conséquence

- L'algorithme utilisé est générique : il n'a pas été utilisé de faiblesse sur le nombre.

Conséquence

- L'algorithme utilisé est générique : il n'a pas été utilisé de faiblesse sur le nombre.
- $10^{250} \approx 830$: Il est possible de factoriser des nombres de l'ordre de 830 bits. En 2010, le record était de 768 bits.
  => Le standard 1024 bits est progressivement remplacé par le 2048 bits.

Algorithme de factorisation et primalité

- Un algorithme A dit de factorisation va en fait trouver un diviseur non-trivial p de N.

Algorithme de factorisation et primalité

- Un algorithme A dit de factorisation va en fait trouver un diviseur non-trivial p de N.
- Il faut savoir quand arrêter.

Algorithme de factorisation et primalité

- Un algorithme A dit de factorisation va en fait trouver un diviseur non-trivial p de N.
- Il faut savoir quand arrêter.
  - Soit lancer l'algorithme A sur p, si il ne renvoie rien c'est que p est premier.
  - Tester avec un algorithme spécifique si p est premier

Algorithme de factorisation et primalité

- Un algorithme A dit de factorisation va en fait trouver un diviseur non-trivial p de N.
- Il faut savoir quand arrêter.
  - Soit lancer l'algorithme A sur p, si il ne renvoie rien c'est que p est premier.
  - Tester avec un algorithme spécifique si p est premier
    Bien plus efficace !
    En pratique on teste systématiquement la primalité avant de lancer un algorithme de factorisation.

Précautions d'usage

Avant de lancer un algorithme de factorisation (potentiellement long) il est préférable d'effectuer quelques tests :

- Vérifier que N n'est pas premier.
- Vérifier que N n'est pas une puissance.
- Vérifier que N ne possède pas un très petit facteur.

Idée

Théorème

Si N est un nombre composé alors il possède un diviseur $\leq \sqrt{N}$.

Idée

Théorème

Si N est un nombre composé alors il possède un diviseur $\leq \sqrt{N}$.

- Idée : tester tous les nombres entre 2 et $\sqrt{N}$.

Idée

Théorème

Si N est un nombre composé alors il possède un diviseur $\leq \sqrt{N}$.

- Idée : tester tous les nombres entre 2 et $\sqrt{N}$.
- Complexité : $O(\sqrt{N})$.

Idée

Théorème

Si N est un nombre composé alors il possède un diviseur $\leq \sqrt{N}$.

- Idée : tester tous les nombres premiers entre 2 et $\sqrt{N}$.

Idée

Théorème

Si N est un nombre composé alors il possède un diviseur $\leq \sqrt{N}$.

- Idée : tester tous les nombres premiers entre 2 et $\sqrt{N}$.

Théorème de répartition des nombres premiers

Pour un réel x, le nombre $\pi(x)$ de nombres premiers inférieurs ou égaux à x est équivalent lorsque x tend vers $+\infty$:

$$\pi(x) \underset{x \to +\infty}{\sim} \frac{x}{\ln(x)}$$

Idée

Théorème

Si N est un nombre composé alors il possède un diviseur $\leq \sqrt{N}$.

- Idée : tester tous les nombres premiers entre 2 et $\sqrt{N}$.
- Complexité : $O(\sqrt{\frac{N}{\ln(N)}})$, le gain est assez faible.

Complexité

- On a un algorithme de complexité $O(\sqrt{N})$. Ça semble extrêmement efficace (sous-linéaire) donc pourquoi le problème de la factorisation est réputé difficile ?

Complexité

- On a un algorithme de complexité $O(\sqrt{N})$. Ça semble extrêmement efficace (sous-linéaire) donc pourquoi le problème de la factorisation est réputé difficile ?
- La complexité s'exprime en fonction de la taille de l'entrée. Le nombre N est stocké sur $\ln(n)$ bits.

Complexité

- On a un algorithme de complexité $O(\sqrt{N})$. Ça semble extrêmement efficace (sous-linéaire) donc pourquoi le problème de la factorisation est réputé difficile ?
- La complexité s'exprime en fonction de la taille de l'entrée. Le nombre N est stocké sur $\ln(n)$ bits.
- En notant $k = \ln(n)$, la complexité s'exprime comme $2^{\frac{k}{2}}$ qui est exponentiel en la taille de l'entrée.

Conclusion

- Algorithme très simple à implémenter.
- Complexité exponentielle.

Conclusion

- Algorithme très simple à implémenter.
- Complexité exponentielle.
- A utiliser uniquement pour des petites valeurs de N.

Conclusion

- Algorithme très simple à implémenter.
- Complexité exponentielle.
- A utiliser uniquement pour des petites valeurs de N.

A titre de comparaison, factoriser RSA-250 avec cette algorithme, à raison de 10^9 opérations par seconde et par ordinateur et 10000 ordinateurs prendrait plus de 10^{100} siècles !

Algorithme

Algorithme 1 : Pollard rho

Input : N, f où f est une fonction pseudo-aléatoire

1 $a \xleftarrow{\$} \mathbb{Z}/N\mathbb{Z}$
2 $(a, b) \leftarrow (a, a)$
3 **while do**
4 $\quad (a, b) \leftarrow (f(a), f(f(b))$
5 $\quad d \leftarrow pgcd(a - b, n)$
6 $\quad$ **if** $1 < d < n$ **then**
$\quad\quad$ **Output :** d
7 $\quad$ **if** $d = n$ **then**
$\quad\quad$ **Output :** erreur

TD 1

Entiers friables

Entier B-friables

Un entier est dit B-friable si tous ses facteurs premiers sont plus petit que B.

Entiers friables

Entier B-friables

Un entier est dit B-friable si tous ses facteurs premiers sont plus petit que B.

Les entiers B-friables sont donc facile à factoriser. Pourtant ce n'est pas ce qui nous intéresse ici...

Un peu d'arithmétique

Petit théorème de Fermat

Soit a un entier et p un nombre premier, alors $a^{p-1} \equiv 1 \mod p$

Un peu d'arithmétique

Petit théorème de Fermat

Soit a un entier et p un nombre premier, alors $a^{p-1} \equiv 1 \mod p$

- Si $p-1$ est B-friable alors $p-1$ divise $M = ppcm(2, ..., B)$. (Attention aux multiplicités !)

Un peu d'arithmétique

Petit théorème de Fermat

Soit a un entier et p un nombre premier, alors $a^{p-1} \equiv 1 \mod p$

- Si $p-1$ est B-friable alors $p-1$ divise $M = ppcm(2, ..., B)$. (Attention aux multiplicités !)
- Dans ce cas $pgcd(n, a^M - 1) \geq p$.

Algorithme (version courte)

- Choisir B et a et calculer $M = \prod_{q \leq B} q^{\log B}$.
- Faire le pgcd de a^M et N.
- Si on n'a pas obtenu un diviseur, recommencer avec un autre B ou changer d'algorithme.

Algorithme (version courte)

- Choisir B et a et calculer $M = \prod_{q \leq B} q^{\log B}$.
- Faire le pgcd de a^M et N.
- Si on n'a pas obtenu un diviseur, recommencer avec un autre B ou changer d'algorithme.

Complexité

La complexité d'une itération à B fixé est :

Algorithme (version courte)

- Choisir B et a et calculer $M = \prod_{q \leq B} q^{\log B}$.
- Faire le pgcd de a^M et N.
- Si on n'a pas obtenu un diviseur, recommencer avec un autre B ou changer d'algorithme.

Complexité

La complexité d'une itération à B fixé est : $O(B \log(B) \log(N)^2)$

Conclusion

- Algorithme spécifique : on peut ne pas obtenir la factorisation.

Conclusion

- Algorithme spécifique : on peut ne pas obtenir la factorisation.
- Si B est de l'ordre de $\log N$ algorithme polynomial !

Conclusion

- Algorithme spécifique : on peut ne pas obtenir la factorisation.
- Si B est de l'ordre de $\log N$ algorithme polynomial !
- Tous les calculs sont bien entendu fait modulo N.

Conclusion

- Algorithme spécifique : on peut ne pas obtenir la factorisation.
- Si B est de l'ordre de $\log N$ algorithme polynomial !
- Tous les calculs sont bien entendu fait modulo N.

Il est important de connaître les algorithmes spécifiques pour ne pas choisir un module RSA de bonne longueur mais vulnérable à ce genre d'attaque.

Identité remarquable

On cherche à écrire N comme la différence de deux carrés :

Principe

$$N = a^2 - b^2 = (a + b)(a - b)$$

Autrement dit

$$a^2 \equiv b^2 \mod N$$

Identité remarquable

On cherche à écrire N comme la différence de deux carrés :

Principe

$$N = a^2 - b^2 = (a + b)(a - b)$$

Autrement dit

$$a^2 \equiv b^2 \mod N$$

Remarque :Tirer des valeurs aléatoires pour a et espérer que $N - a^2$ mod N soit un carré n'est pas efficace.

Criblage

Schéma d'algorithme en deux phases

- Une première phase de collecte de relations (hautement parallélisable).
- Une seconde phase d'exploitation pour trouver une factorisation (souvent à base l'algèbre linéaire).

Remarque : la plupart de ces algorithmes s'adaptent très facilement pour le calcul du logarithme discret.

Principe

- Pour obtenir une congruence de carré on cherche à obtenir des relations de factorisation plus faibles.

Principe

- Pour obtenir une congruence de carré on cherche à obtenir des relations de factorisation plus faibles.
- On fixe un entier B et on cherche des entiers x tels que x^2 soit B-friable. Relations : $x^2 = \prod p_j^{u_i j}$

Principe

- Pour obtenir une congruence de carré on cherche à obtenir des relations de factorisation plus faibles.
- On fixe un entier B et on cherche des entiers x tels que x^2 soit B-friable. Relations : $x^2 = \prod p_j^{u_i j}$
- Résoudre le système matriciel $(u_{ij})^t.v = 0 \mod 2$

Principe

- Pour obtenir une congruence de carré on cherche à obtenir des relations de factorisation plus faibles.
- On fixe un entier B et on cherche des entiers x tels que x^2 soit B-friable. Relations : $x^2 = \prod p_j^{u_i j}$
- Résoudre le système matriciel $(u_{ij})^t.v = 0 \mod 2$
- $Y = \prod_i x_i^{v_i} \mod N$ et $Z = \prod x_i^{\sum_j v_{ij} v_i}$ sont candidats.
- Si $Y \neq \pm Z$ renvoyer le pgcd de N et $Y - Z$.

Exemple

- On fixe un entier B et on cherche des entiers x tels que x^2 soit B-friable. Relations : $x^2 = \prod p_j^{u_i j}$
- $N = 84923$, $B = 7$

$$513^2 \equiv 2^4.3.5^2.7 \mod N$$

$$537^2 \equiv 2^6.3.5^2.7 \mod N$$

Exemple

- $N = 84923,\ B = 7$

$$513^2 \equiv 2^4.3.5^2.7 \mod N$$

$$537^2 \equiv 2^6.3.5^2.7 \mod N$$

- Résoudre le système matriciel $(u_{ij})^t.v = 0 \mod 2$

$$\begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} .(v_1, v_2) = 0 \Longrightarrow v = (1, 1)$$

Exemple

- $N = 84923,\ B = 7$

$$513^2 \equiv 2^4.3.5^2.7 \mod N$$
$$537^2 \equiv 2^6.3.5^2.7 \mod N$$
$$v = (1, 1)$$

- $Y = \prod_i x_i^{v_i} \mod N$ et $Z = \prod x_i^{\sum_j v_{ij} v_i}$ sont candidats.

$$Y^2 = (513 * 537)^2 \mod N = 20712^2 \mod N$$
$$Z^2 = 2^{10}.3.5^4.7 = (2^5.3.5^2.7)^2 = 16800^2 \mod N$$

Exemple

- $N = 84923,\ B = 7$

$$513^2 \equiv 2^4.3.5^2.7 \mod N$$
$$537^2 \equiv 2^6.3.5^2.7 \mod N$$
$$v = (1, 1)$$

$$Y^2 = (513 * 537)^2 \mod N = 20712^2 \mod N$$
$$Z^2 = 2^{10}.3.5^4.7 = (2^5.3.5^2.7)^2 = 16800^2 \mod N$$

- Si $Y \neq \pm Z$ renvoyer le pgcd de N et $Y - Z$.
- On trouve comme pgcd le facteur 163 !

Complexité

- Si B est trop petit, difficile d'obtenir des relations (phase 1 longue).
- Si B est trop grand, le système matriciel sera long à résoudre (phase 2 longue)

Complexité

- Si B est trop petit, difficile d'obtenir des relations (phase 1 longue).
- Si B est trop grand, le système matriciel sera long à résoudre (phase 2 longue)
- Choix de B : de l'ordre de $\exp(\sqrt{\log N \log\log N})$.
- Complexité : $O(\exp 2\sqrt{2}(\sqrt{\log N \log\log N}))$

Complexité

- Tous les algorithmes vus ont une complexité qui peut être mise sous la forme $O(\exp c((\log N)^{\alpha}(\log\log N)^{1-\alpha}))$.

Complexité

- Tous les algorithmes vus ont une complexité qui peut être mise sous la forme $O(\exp c((\log N)^{\alpha}(\log\log N)^{1-\alpha}))$.
- $\alpha = 0$: algorithme polynomial : on n'en connaît pas dans le cas général.

Complexité

- Tous les algorithmes vus ont une complexité qui peut être mise sous la forme $O(\exp c((\log N)^{\alpha}(\log\log N)^{1-\alpha}))$.
- $\alpha = 0$: algorithme polynomial : on n'en connaît pas dans le cas général.
- $\alpha = 1$: algorithme exponentiel : division naïve, Pollard-rho. Inefficace pour des grandes valeurs de N, en revanche assez simple à mettre en place.

Complexité

- Tous les algorithmes vus ont une complexité qui peut être mise sous la forme $O(\exp c((\log N)^{\alpha}(\log\log N)^{1-\alpha}))$.
- $\alpha = 0$: algorithme polynomial : on n'en connaît pas dans le cas général.
- $\alpha = 1$: algorithme exponentiel : division naïve, Pollard-rho. Inefficace pour des grandes valeurs de N, en revanche assez simple à mettre en place.
- α intermédiaire : algorithme de criblage, $\alpha = \frac{1}{2}$ pour Dixon. Le crible quadratique améliore la constante c en $c = 1$. Important car la constante est dans l'exponentielle.
- Complexité (heuristique) de NFS : $\alpha = \frac{1}{3}$, $c \approx 1,922$.

Algorithme de Shor

Théorème

L'algorithme de Shor factorise un entier N en temps $O(\log N)^3$.

"Détail" : il s'agit d'un algorithme quantique !

Algorithme de Shor

Théorème

L'algorithme de Shor factorise un entier N en temps $O(\log N)^3$.

"Détail" : il s'agit d'un algorithme quantique !

=> Les systèmes basés sur RSA ne résistent pas à une démocratisation de l'ordinateur quantique.