Um novo algoritmo probabilístico para fatoração de inteiros com primos relativamente distantes

Fábio Borges[1]

[1]Coordenação de Sistemas e Redes – Laboratório Nacional de Computação Científica
Av. Getúlio Vargas, 333, CEP: 25651-075 Petrópolis - Rio de Janeiro.

borges@lncc.br

1. Introdução e Trabalhos Relacionados

Muitos algoritmos criptográficos têm baseado sua segurança no Problema da Fatoração de Inteiros (PFI), desta forma, tal problema tem sido muito importante para a segurança da informação e de sistemas computacionais. Este trabalho apresenta um novo algoritmo probabilístico para fatorar $n = pq$ quando p e q são primos relativamente distantes.

Atualmente, a segurança de muitos sistemas computacionais é garantida pelo PFI, na grande maioria dos casos através do uso do algoritmo RSA. O PFI reside na dificuldade de determinar primos $p_1 \dots p_i$ e inteiros $e_1 \dots e_i$, tais que $n = p_1^{e_1} \cdots p_i^{e_i}$. Em criptografia o PFI é reduzido a encontrar dois primos grandes p e q dado o produto $n = pq$. Se os números forem pequenos, torna-se fácil resolver o PFI, para fornecer uma boa margem de segurança normalmente n tem 1024 bits, ou seja, 309 dígitos em decimal [Stallings 2002]. Existem diversos algoritmos de fatoração, porém, todos têm complexidade exponencial. No entanto, caso seja feita uma má escolha dos primos, temos algoritmos que podem fornecer-los rapidamente. O conhecido método de Fermat foi o primeiro a nos alertar que não podemos escolher quaisquer primos p e q, pois se eles forem suficientemente próximos podemos encontrá-los facilmente. Pollard mostrou que cada primo p_i de n tem que ter mais uma restrição, $p_i - 1$ deve ter fatores primos grandes. O melhor caso para a segurança acontece quando $(p_i - 1)/2$ for um número primo. Posteriormente, Lenstra generalizou o método $p_i - 1$ de Pollard para trabalhar no grupo $\Omega(\mathbb{Z}_p)$, formado pelos pontos de uma curva elíptica. É relevante enfatizar que os algoritmos de fatoração citados não são viáveis para fatorar todos inteiros. No entanto, se suas condições são satisfeitas temos algoritmos muito eficientes. Atualmente, a melhor estratégia para a fatoração de um inteiro de tamanho arbitrário é o uso seqüencial de alguns algoritmos. Até onde sabemos a melhor abordagem foi proposta por [Brent et al. 2000], onde se usa o método de Lenstra para a obtenção de fatores com menos de 30 dígitos e depois usa-se o *multiple polynomial quadratic sieve* (MPQS) e o *general number field sieve* (GNFS) para encontrar fatores maiores. Um bom *survey* sobre métodos modernos de fatoração pode ser encontrado em [Lenstra 2000].

2. Fatorando inteiros com primos distantes

Suponha $p < q$, então note que $pq \mod q - 1 = p(q-1) + p \mod q - 1 = p$. Generalizando temos $p(q-s)+sp \mod q-s = sp$ sempre que $sp < q-s$. Podemos fatorar procurando um número $m = q-s$, tal que $\mathrm{mdc}\,(pq \mod m, pq) > 1$, por exemplo, se $m = 139$ então $\mathrm{mdc}\,(0, 7 \cdot 139) = 973$, mas se $m = 138$ então $\mathrm{mdc}\,(7 \cdot 139 \mod 138, 7 \cdot 139) = 7$. Como os vários possíveis valores de m que possibilitam a fatoração estão próximos, devemos fazer uma busca de forma randômica. Por exemplo, $973 = 7 \cdot 139$ tem 17 soluções

consecutivas entre 121 e 139, sendo 166 soluções no total, assim, em uma busca aleatória temos mais chances de encontrar uma das 17 soluções do que em uma busca seqüencial. Poderíamos fazer m variar de 2 até pq, mas é recomendado o algoritmo 1 para aumentar a probabilidade de encontrarmos um fator de n, desta forma a busca não é seqüencial. Os parâmetros do algoritmo 1 estão ajustados para primos não tão distantes. Por exemplo, se $p = 997\,845\,647$ e $q = 1\,134\,515\,747$ o algoritmo efetua 149 095 241 passos, estamos medindo os passos pelo número de vezes que a condição IF foi verificada. Dependendo de n, aumentar o espaço de busca no primeiro laço pode diminuir o número de passos. No caso de $n = 7 \cdot 139$, estes parâmetros não são uma boa escolha, pois geram 8 passos. No entanto, se fizermos m variar de r até n podemos encontrar o fator 7 em 3 passos.

Algoritmo 1 Encontrando fatores distantes

```
1. i = 1; v = 1; r = [√n];
2. while mdc (v, n) = 1 and m < 3r do
3.         i = nextprime(i);
4.         for m from r to 3r by [3r/i] do
5.                 v = n mod m;
6.                 if (mdc (v, n) > 1) then Return(mdc (v, n));
```

Vejamos a relação entre s e a distância $d = q - p$, como $p(d + p - s) + sp \mod d+p-s$ se $sp < d+p-s$ então $p(s-1)+p < d-s+p$, isto é $p(s-1) < d-s$. Se nós considerarmos s como uma variável então a curva da esquerda tem um coeficiente, assim $p(s-1)$ cresce mais rápido que a curva $d-s$. Nós podemos ver que $s(p+1)-s < d-s+p$, logo $s(p+1) < d+p$, ou seja, $s(p+1) < q$. Portanto, $s < \frac{q}{p+1}$. Este método trabalha no lado oposto do método de Fermat, ou seja, quando p e q são suficientemente distantes. Toda a velocidade do algoritmo depende de s, que depende da escolha dos primos.

A idéia central do algoritmo é encontrar m tal que $\mathrm{mdc}\ (pq \mod m, pq) > 1$. Observe que $pq \mod m \neq ((p \mod m) \cdot (q \mod m)) \mod m$, isto é, nem sempre se verifica uma igualdade, mas nos possibilita aumentar as chances de encontrarmos um fator de n, pois conforme os primos se distanciam temos maior chances de encontrar uma igualdade. Assim o algoritmo poderia usar diversas formas pseudo-aleatórias para buscar tal igualdade.

3. Conclusão

Quanto maior for s, maior a distância d e melhor será o algoritmo. Assim, temos um novo algoritmo probabilístico que fatora inteiros com primos suficientemente distantes. A eficiência do algoritmo depende diretamente da escolha dos primos e indiretamente do espaço de busca escolhido.

Referências

Brent, R., Montgomery, P., and Riele, H. (2000). Factorizations of cunningham numbers with bases 13 to 99: Millenium edition.

Lenstra, A. K. (2000). Integer factoring. *Des. Codes Cryptography*, 19(2-3):101–128.

Stallings, W. (2002). *Cryptography and Network Security: Principles and Practice*. Pearson Education, third edition.