INTRODUÇÃO À CRIPTOGRAFIA E AO CÓDIGO RSA

Aluno: Guilherme Frederico Lima
Orientador: Derek Hacon

Introdução

A segurança do sistema RSA reside no fato de que certas operações são fáceis de serem feitas mas difíceis de serem desfeitas. Mas como distinguir de forma rigorosa e, portanto, segura o fácil do difícil neste caso? Esta pergunta pertence a ramo da ciência da computação denominado teoria da complexidade computacional; e, de fato, já possui uma resposta razoavelmente satisfatória porém algumas questões de extrema importância ainda estão em aberto.

Algoritmos

A complexidade de uma dada operação é calculada a partir dos recursos requeridos pelo algoritmo que utilizamos. Na maioria das vezes, os recursos mais relevantes são o tempo e o espaço. Por algoritmo entende-se um procedimento mecânico que não requer nenhuma engenhosidade daquele que o executa. Em 1936, Alan Turing propôs uma noção formal de algoritmo conhecida como máquina de Turing.

Uma máquina de Turing (ou máquina de Turing determinística) nada mais é do que uma função que, ao receber uma seqüência finita de símbolos (geralmente números em binário representados por zeros e uns), executa uma seqüência de operações, podendo tanto alterar a seqüência original, responder "sim" ou "não", ou nunca parar de calcular. Turing conjeturou que a sua máquina poderia executar qualquer algoritmo que um ser humano é capaz de efetuar, e até hoje não se conhece computador que consiga transcender a capacidade de cálculo de uma máquina de Turing e cuja velocidade não difira polinomialmente desta.

Um outro modelo computacional importante porém inverossímil e de valor puramente teórico é o da máquina de Turing não-determinística. Neste caso, os algoritmos não são efetuados passo a passo, mas em paralelo, como se a máquina efetuasse vários procedimentos simultâneos.

Classes de Complexidade

Dizemos que um dado algoritmo é rápido se o tempo de sua execução pode ser estimado superiormente por uma função polinomial, e lento se tal função for exponencial. Normalmente associamos uma unidade de tempo a cada passo de uma máquina de Turing.

A classe de problemas ou operações que uma máquina de Turing decide ou efetua em tempo polinomial é denominada P, e a classe de problemas ou operações que uma máquina de Turing não-determinística decide ou efetua em tempo polinomial de NP. Informalmente, podemos compreender a classe P como aquela cujos problemas ou operações sabemos calcular diretamente; e a classe NP aquela cujos problemas ou operações não sabemos como resolver diretamente mas somos capazes de tentar "adivinhar" uma resposta e rapidamente testá-la para saber se está correta ou não.

Sabemos que é possível construir uma máquina de Turing determinística que resolve um problema NP, porém o tempo que não será polinomial, e sim exponencial. Isso se dá em virtude do processo de tentativa e erro que a máquina efetua.

Um problema que se encontra ainda em aberto é o famoso P versus NP, que pergunta se para todo problema NP não haveria um algoritmo P ainda a ser descoberto. Uma das

classes de complexidade mais relevantes neste caso é a NP-completa. Um problema é NP-completo se todo outro problema NP pode ser reduzido a ele em tempo polinomial. Achar um algoritmo P para um problema NP-completo significa achar um algoritmo rápido para todo problema, e isto implicaria que P = NP, o que traria conseqüências desastrosas à criptografia de chave pública.

Criptografia de Chave Pública

A idéia por trás da criptografia de chave pública é que, mesmo que a mensagem seja interceptada e que todos saibam como codificar o texto, apenas o destinatário seria capaz de decodificá-la. Isto seria possível devido às assimetrias presentes entre certas operações e suas inversas. Em particular, o sistema RSA usa duas assimetrias: a da multiplicação e fatoração; e a da exponenciação e radiciação, sendo esta última no contexto especial de um corpo finito. Olhando para o primeiro caso, é fácil ver que para a multiplicação possuímos um algoritmo direto, mas, quando desejamos fatorar um número, prosseguimos testando uma divisão por primos em ordem crescente. Vemos que multiplicação de inteiros é P e fatoração é NP.

O Sistema de Criptografia RSA

O sistema RSA codifica uma mensagem elevando ela a um expoente *e* e achando seu resíduo módulo n, isto é: $C(M) = M^e \equiv M_c \pmod n$. Para decodificar, ela eleva a mensagem codificada a outro expoente *d* e novamente acha seu resíduo módulo n: $D(M_c) = M^d \equiv M \pmod n$. É evidente que para isto ocorrer *e* precisa ser algo semelhante ao inverso multiplicativo de *d* neste sistema de congruência. Pelo Teorema de Euler-Fermat temos que: Se mdc(a,n) = 1, então $a^{\phi(n)} \equiv 1 \pmod n$. Portanto: $1 \equiv a^{\phi(n)} \equiv a^{K\phi(n)} \pmod n \Rightarrow a \equiv a^{K\phi(n)+1} \pmod n$. Logo $ed = K\phi(n) + 1$. Pelo Teorema Chinês do Resto sabemos que para todo *e* coprimo a $\phi(n)$ poderemos achar um *d* que satisfaça esta equação. Porém, para calcular $\phi(n)$ precisamos saber sua decomposição em fatores primos. Se quem recebe a mensagem escolhe n como sendo o produto de dois primos grandes, sua fatoração se torna impraticável. Outra forma de tentar quebrar o sistema é extraindo a raiz *e*-ésima de M^e, mas a forma aparentemente aleatória que as potências de um elemento em um corpo finito se distribuem também torna tal método impraticável. Portanto, os números *e* e n não precisam ser secretos, pois apenas quem conhece a fatoração de n é capaz de descobrir *d*, e decifrar a mensagem. Aquele que deseja receber uma mensagem divulga *e* e n para que aquele que for enviar a mensagem possa criptografá-la, porém fatorar n em tempo hábil é impossível, e portanto somente o destinatário conhece *d*, e pode decodificar a mensagem.

Referencias Bibliográficas

1 - KOBLITZ, N. **Algebraic aspects of cryptography**. 1.ed. Nova Iorque: Springer-Verlag, 1999. 206p.

2 - PAPADIMITRIOU, C. H. **Computational complexity**.1.ed. Adison-Wesley Publishing Company, 1994. 523p.

3 - GAREY, M. R., JOHNSON, D. S. **Computers and intractability: a guide to the theory of NP – Completeness.** Nova Iorque: W. H. Freeman, 1979. 340p.

4 – GRAHAM, D. R., KNUTH, D. E., PATASHNIK, O. **Matemática concreta – fundamentos para a ciência da computação.** 2ed. Rio de Janeiro: Livros Técnicos e Científicos, 1995. 475p.