UNIVERSIDADE FEDERAL DE JUIZ DE FORA
Instituto de Ciências Exatas
Departamento de Matemática

A matemática dos testes de primalidade

Professora orientadora: Beatriz Casulari da Motta Ribeiro (beatriz@ice.ufjf.br)
Aluno: Danilo Machado Tereza (danilomachadot@gmail.com)
Área: Primária MSC11-04, Secundárias 11Y05, 11Y11, 11Y16

Resumo

A criptografia de chave pública é uma classe de protocolos de criptografia baseados em problemas matemáticos que atualmente não admitem solução eficiente. Em geral, requerem um par de chaves, uma delas sendo secreta e a outra delas sendo pública, que, apesar de diferentes, são matematicamente relacionadas. É computacionalmente fácil para um usuário gerar um par de chaves, uma pública e uma privada, e usá-lo para encriptação e decriptação. A segurança está na "impossibilidade" (computacionalmente) para uma chave privada gerada apropriadamente ser determinada pela sua chave pública correspondente. No tipo mais clássico de criptografia de chave pública, a segurança se baseia, basicamente, na dificuldade computacional da decisão sobre a primalidade ou não de números inteiros muito grandes.

Um teste de primalidade refere-se a um algoritmo que tendo como entrada um inteiro positivo n, determina se n é ou não primo. Há basicamente dois grandes grupos de algoritmos: os determinísticos e os não determinísticos. Os algoritmos do primeiro grupo determinam com exatidão a primalidade de um número, porém, na maioria dos casos, seu custo aumenta muito conforme aumentamos o número estudado. Em paralelo, estão os do segundo grupo, que, embora sejam menos custosos, não determinam exatamente se o número é primo.

Nesse projeto de iniciação científica, estudamos a matemática por trás de três algoritmos de primalidade nos baseando especialmente na referência [1]: o primeiro determinístico com custo exponencial; já o segundo com custo polinomial pertence ao grupo dos não determinísticos, e consegue determinar com exatidão apenas se o número é composto; por fim, estudamos o teste AKS, primeiro algoritmo simultaneamente polinomial, determinístico, e incondicional.

Para os estudos dos citados algoritmos, foi necessário o estudo da base matemática: artimética dos inteiros, grupos abelianos e anéis comutativos. Na teoria de grupos abelianos estudada, o objetivo era o teorema de Lagrange, que afirma que a ordem de um grupo finito é divisível pela ordem de qualquer um dos seus subgrupos. Além do fato de que se G é um grupo finito e $a \in G$, então um inteiro positivo t satisfaz $a^t = e$ se, e somente se, t é divisível pela ordem de a. Já, na teoria de aneis comutativos, o objetivo era compreender a fatoração única de polinômios para assim enunciar o seguinte resultado: um polinômio de grau n, com coeficientes em um corpo K, não pode ter mais do que n raízes em K. Ainda foi necessário o seguinte resultado, corolário do Pequeno Teorema de Fermat:

Lema 1. *Sejam n e b inteiros positivos e primos entre si. Então, $(x + \bar{b})^n = x^n + \bar{b}$ em $\mathbb{Z}_n[x]$ se, e somente se, n é primo.*

Esse lema é a chave do primeiro teste determinístico de primalidade estudado, que tem como entrada um inteiro positivo n e retorna *primo* ou *composto*, executando apenas as seguintes etapas:

Etapa 1: Calcule todos os termos da expressão $(x + \bar{b})^n$ em $\mathbb{Z}_n[x]$.

Etapa 2: Verifique se a expressão obtida na etapa anterior é igual a $x^n + \bar{b}$. Se for, retorna *primo*, caso contrário, retorna *composto*.

Como estamos em $\mathbb{Z}_n$, temos um controle maior das nossas operações ao longo do algoritmo, isto é, os valores de todas as variáveis no algoritmo estão sendo processadas módulo n. É claro que este controle foge conforme o n cresce muito, exigindo assim, uma alocação maior de memória. Como o algoritmo procura sistematicamente a validade na igualdade entre os valores primos com n, seu tempo de processamento é, em média, exponencial. Portanto, mesmo o algoritmo sendo exato na resposta, ele se torna impraticável para valores maiores de n.

A recíproca do Teorema de Fermat, isto é, se $b^{n-1} \equiv 1 \pmod{n}$ para todo $1 \leqslant b \leqslant n-1$, então n é primo, em geral é falsa. Os números conhecidos como *números de Carmichael* são aqueles que satisfazem a hipótese dessa recíproca, porém não são primos. Essa é a chave para o teste não-determinístico estudado em seguida, que apenas determina com exatidão quando o número é composto. Isso significa que, em certos casos, o procedimento determinaria, por exclusão, que um algum número composto seria primo.

A entrada do teste são inteiros n e b, tais que $1 < b < n-1$, retornando n *é composto* ou *teste inconclusivo*, após a excecução das seguintes etapas:

Etapa 1: Divida $n-1$ sucessivamente por 2 até encontrar q (ímpar) e k tais que $n-1 = 2^k q$.

Etapa 2: Comece tomando $i = 0$ e r o resíduo de b^q por n.

Etapa 3: Se $i = 0$ e $r = 1$ ou se $i \geqslant 0$ e $r = n-1$ retorne *teste inconclusivo.*

Etapa 4: Incremente i em 1 e substitua r pelo resto da divisão de r^2 por n.

Etapa 5: Se $i \leqslant k-1$ volte à etapa 3, senão retorne *composto.* caso contrário, retorna *composto.*

Nesse teste, é reduzido o tempo de processamento que, em média, torna-se polinomial. A exigência computacional também não é grande, pois trabalhamos com operações na ordem do valor de n, sendo então o "único" problema a falta de exatidão na resposta.

Chegamos enfim ao estudo do algoritmo AKS, que se baseia na equivalência $(x-a)^n \equiv (x^n - a) \pmod{n}$, a qual é verdadeira somente se n é primo, que é uma generalização do Teorema de Fermat estendido para polinômios. Porém, observamos anteriormente que essa verificação constitui um teste de primalidade por si só, em tempo exponencial. Além disto AKS faz uso da relação $(x-a)^n \equiv (x^n - a) \pmod{n}$ em $\mathbb{Z}_p[x]/(x^r-1)$. Contudo, embora todos os primos satisfaçam a relação, alguns números compostos também o fazem. Assim, é necessário mostrar que existe um conveniente menor r e um conveniente conjunto $A \subset \mathbb{Z}$ tais que se a equivalência é cumprida por todo $a \in A$, então n é primo.

Teorema 2. *Sejam $n \geqslant 2$ e $r \geqslant 1$ inteiros positivos e seja $S = \{1, 2, \cdots, r\}$. Suponha que:*

1. *r é um primo que não divide n;*
2. *$mdc(n, b-b') = 1$ quaisquer que sejam b e b' distintos em S;*
3. *a desigualdade $\binom{2r-2}{r} \geqslant n^{2d\lfloor\sqrt{(r-1)/d}\rfloor}$ vale para qualquer inteiro positivo d que divide $(r-1)/v$, onde v é a ordem de n em $U(r)$; e*
4. *a igualdade $(\hat{x} + \hat{b})^n = \hat{x}^n + \hat{b}$ é verificada em $\mathbb{Z}_p[x]/(x^r-1)$ para cada $b \in S$*

Então, n é uma potência de um primo.

O algoritmo AKS baseia-se no Teorema 2 e consiste de duas partes. O primeiro passo é encontrar um primo conveniente $r = kq + 1$ que satisfaz algumas condições. Durante este passo, é importante confirmar que n não é divisível por nenhum primo $p \leq r$, caso contrário n é composto. No segundo passo, são feitos teste em um corpo finito. A entrada do algoritmo é um inteiro positivo ímpar n e a saída é *primo* ou *composto*, executando as seguintes etapas:

Etapa 1: Verifique se n é uma potência de um inteiro. Se for, retorne composto e encerre o algoritmo.

Etapa 2: Se não, calcule $N = 2n(n-1)(n^2-1)(n^3-1)\cdots(n^{4\lceil \log_2 n \rceil^2}-1)$ e determine o menor primo r que não divide N.

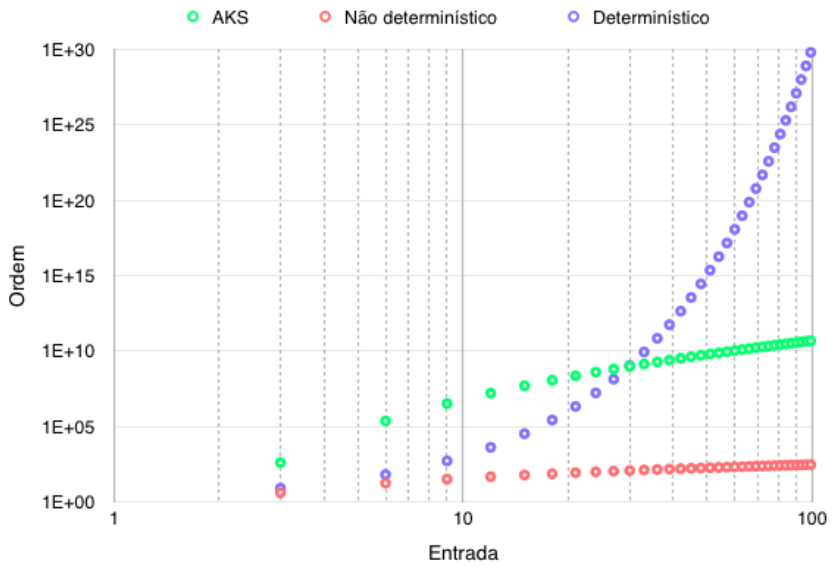
Etapa 3: Verifique se n é divisível por algum $q < r$. Se for e se $q = n$ então retorne primo e encerre, mas se $q < n$ então imprima composto e encerre.

Etapa 4: Caso contrário, verifique se $(\hat{x}+\hat{b})^n = \hat{x}^n + \hat{b}$ em $\mathbb{Z}_n[x]/(x^r-1)$ para todo $1 \leq b \leq r$. Se a congruência não for verificada para algum $1 \leq b \leq r$, retorne composto e encerre.

Etapa 5: Se a congruência for verificada para todo $1 \leq b \leq r$, retorne primo e encerre.

Assim, algebricamente, o AKS une o melhor dos testes anteriores. Porém, exige um cuidado maior com a alocação de memória, principalmente para o cálculo de N.

Uma análise mais cuidadosa do custo dos algoritmos mostra que o teste determinístico tem ordem $\mathcal{O}(2^k)$, onde k é o número de bits da entrada n, confirmando seu custo exponencial. Por outro lado, o teste não determinístico tem ordem $\mathcal{O}((log_2 n)^3)$ e o AKS tem ordem $\mathcal{O}((log_2 n)^3 3)$, onde n é a entrada, confirmando assim seus custos polinomiais. A seguir, apresentamos um comparativo do custo de execução de cada algoritmo com relação à entrada n.



Ao longo desse estudo, vimos que a primalidade de um número é, em geral, difícil de ser verificada, já que a ideia intuitiva torna-se difícil para valores ainda pequenos e, quando se trabalha resultados mais refinados, devemos construi-los equilibrando a precisão matemática com o custo de processamento. Assim, o estudo matemático por trás dos testes de primalidade é fundamental para um avanço mais aprofundado na área e um aperfeiçoamento dos algoritmos já existentes.

Referências bibliográficas:

[1] S. C. Coutinho. Primalidade em Tempo Polinomial: Uma introdução ao Algoritmo AKS. *Coleção Iniciação cientifica*, Rio de Janeiro: SBM, 2004.

[2] S. C. Coutinho. Números Inteiros e Criptografia RSA. *Coleção Matemática e Aplicações*, Rio de Janeiro: IMPA, 2013.

[3] A. Gonçalves. Introdução à Álgebra. 5^a edição. *Coleção Projeto Euclides*, Rio de Janeiro: IMPA, 2015.