Complexidade de Algoritmos

Prof. Rafael Alceste Berri
rafaelberri@gmail.com

Prof. Diego Buchinger
diego.buchinger@outlook.com

Prof. Cristiano Damiani Vasconcellos
cristiano.vasconcellos@udesc.br

Complexidade, Criptografia e um pouco de Teoria dos Números

Criptografia RSA

Ideia: Permitir que comunicação entre dois participantes sem que um intruso possa entender as mensagens trocadas.

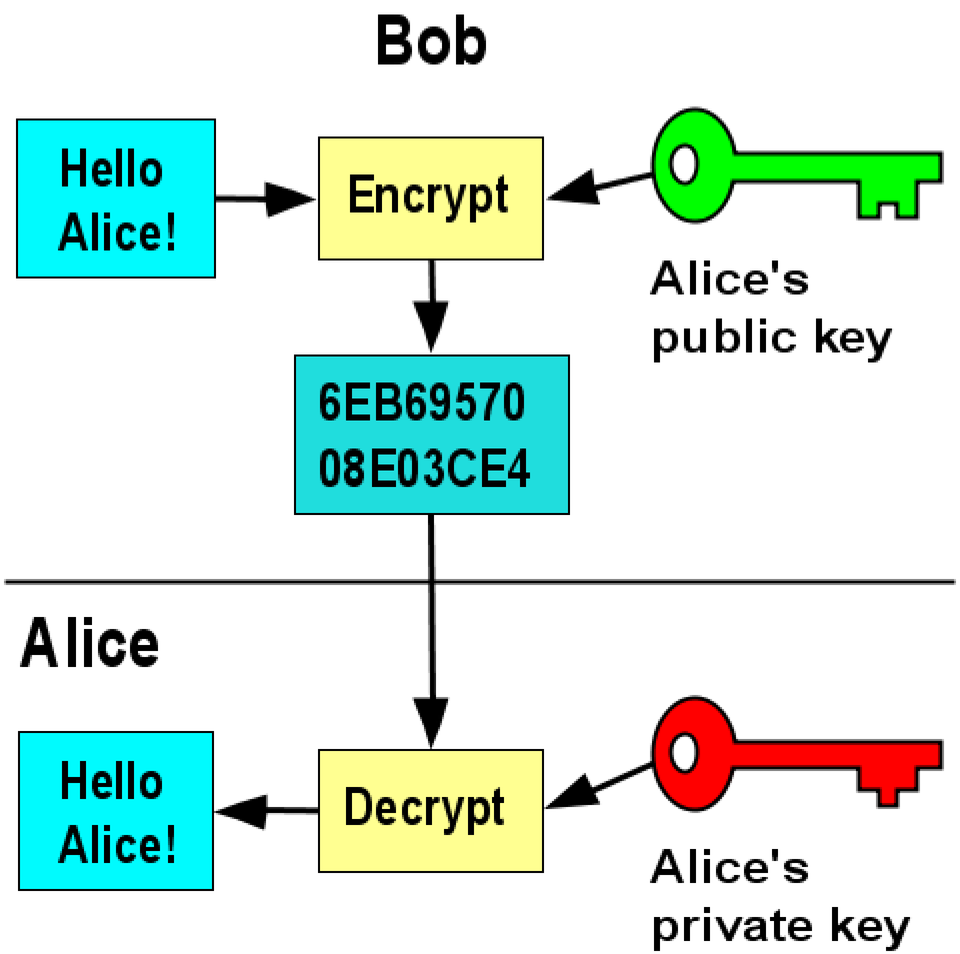
Baseia-se na facilidade em se encontrar números primos grandes e na dificuldade em fatorar o produto entre dois números primos grandes.

Em um sistema de criptografia de chave pública, cada participante possui:

+ uma **chave pública** (pública);

+ uma **chave privada** (secreta);

Criptografia RSA



Como funciona:

- Bob obtém a chave pública de Alice;

- Bob usa a chave para codificar a mensagem M: $C = P_A(M)$ e envia C;

- Alice recebe C e utiliza sua chave privada para recuperar a mensagem original M: $M = S_A(C)$

Criptografia RSA (1)

Algoritmo:

- Selecionar dois números primos grandes ***p*** e ***q***, (512 bits, cada por exemplo) sendo $p \neq q$
- Calcular: $\boldsymbol{n = p * q}$
- Selecionar um inteiro ímpar "pequeno" ***e*** tal que ***e*** seja primo relativo de $(p - 1)(q - 1)$ [número primo]:

$$mdc\,(e, (p - 1)(q - 1)) = 1$$

Chave pública = (*e*, *n*)

O Caráter Primo

Notação: d | a ➔ d "divide" a

sendo que d ≥ 1 e d ≤ |a|

- Todo inteiro a é divisível pelos **divisores triviais** 1 e a
- <u>Número primo</u>: únicos divisores são 1 e a
- Todo número composto pode representado pela multiplicação de seus fatores primos (42 = 2*3*7)
- Qual a complexidade de tempo para descobrir se um número a qualquer (**int**) é primo?

O Caráter Primo

Densidade de números primos

Distribuição dos primos: $\pi(n)$ → *nº de primos* $\leq n$

Exemplo: $\pi(10) = 4$ → *{2,3,5,7}*

Teorema dos números primos: $$\lim_{n\to\infty} \frac{\pi(n)}{n / \ln n} = 1$$

Para *n* grande, *n* / ln *n* é uma boa aproximação para $\pi(n)$!

O Caráter Primo

Densidade de números primos

Com base no teorema apresentado podemos fazer uma estimativa de probabilidade para verificar se um número escolhido ao acaso é primo ou não como: $1 / \ln(n)$

Quantos números de 512 bits precisamos testar, em média, até encontrar um número primo?

$\ln(2^{512}) \approx 355$ números

$1 / 355 \approx 0{,}28\%$ (chance de encontrar um primo de 1ª)

O Caráter Primo

Densidade de números primos

Para testarmos o caráter primo de um número pequeno n, podemos testar verificando a divisibilidade por todos os números entre **2** e $\sqrt{\boldsymbol{n}}$

Para inteiros pequenos: $\Theta(\sqrt{n})$

Para inteiros grandes com k bits: $\Theta(2^k)$

Crivo de eratóstenes

- O crivo de eratóstenes (algoritmo) é um dos mais conhecidos para criar uma lista de primos até um dado *n*.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

Atividade 08

Crie um algoritmo baseado no crivo de eratóstenes e apresente, como resposta, todos os números primos entre 1 e 500. Qual é a complexidade de tempo e espaço esperada para o algoritmo?

(https://pt.wikipedia.org/wiki/Crivo_de_Erat%C3%B3stenes)

Envie os exercícios para rafaelberri@gmail.com Assunto: "TC-CAL08"
Anexar: resultados/código fonte

Teoria dos Números

Teorema de Fermat:

Se p é primo então:

$$a^{p-1} \equiv 1 \ (mod\ p)$$

Aritmética modular
(ver próximo slide)

Contudo, a^{p-1} pode ser um número relativamente grande, e realizar a operação de módulo pode ser um problema.

Calcular:

a=2 / p=5

a=5 / p=3

a=4 / p=7

Aritmética Modular

É um sistema para manipular faixas restritas de números inteiros.

Relação de congruência:

$a \equiv b \ (mod\ n)$ se e somente se a mod $n = b$ mod n.

$a \equiv b \ (mod\ n) \Leftrightarrow n\ divide\ (a - b)$.

Exemplos:

$38 \equiv 14 \ (mod\ 12)$, 38 mod 12 = 14 mod 12

$-10 \equiv 38 \ (mod\ 12)$, -10 mod 12 = 38 mod 12

Exponenciação Modular:

Realizar a operação de elevação ao quadrado repetida e realizar o módulo sempre possível.

Exemplo: $2^{53}\ mod\ 101$

Teste do caráter pseudoprimo

Considerando novamente a equação modular:

$$a^{n-1} \equiv 1 \ (mod\ n)$$

O teorema de Fermat nos diz que se n é primo, então n satisfaz esta equação para qualquer escolha de a ($a \in Z^{+}{}_{n}$)

Se encontrarmos um a que não satisfaça a equação, então certamente n não é primo

Teste do caráter pseudoprimo

Ao testarmos se: $2^{n-1} \equiv 1 \ (mod\ n)$

caso **falso**: *n* certamente não é primo

caso **verdade**: ou *n* é primo ou *n* é pseudoprimo de base 2

Mas, com que frequência há um falso positivo?

Raramente! Existem apenas 22 valores menores que 10.000: {341, 561, 645, 1105, ...}

Usando 512 bits a chance é de $1 / 10^{20}$

Usando 1024 bits a chance é de $1 / 10^{41}$

Teste do caráter pseudoprimo

Teste Aleatório do Caráter primo de Miller-Rabin

- Experimentar diversos valores como base:

  Melhora a confiabilidade, mas existem números “traiçoeiros” e extremamente raros que dão falso positivo para diferentes bases (números de Carmichael)

MILLER-RABIN(n, s) [n é inteiro, ímpar, n ≥ 3]

Escreve **n − 1** na forma $2^k m$ [com k máximo e m maior ímpar]

para j=1 até s

a = RANDOM( ) [valores possíveis: 2 ≤ a ≤ n − 2]

se (!WITNESS(a, n, k, m))

retorne falso [número composto, certamente]

retorne verdade [número quase certamente primo]

Teste do caráter pseudoprimo

Teste Aleatório do Caráter primo de Miller-Rabin

❖ Nossa testemunha (witness) a de que n é composto (com certeza) ou primo com alguma pequena dúvida (pseudoprimo).

WITNESS(a, n, k, m) [n é inteiro, ímpar, n $\geq$ 3]
b $\leftarrow$ a^m mod n
se (b = 1 ou b = n-1)
returne verdade [pseudoprimo]
para j=1 até k - 1
se (b = −1 ou b = n-1)
retorne verdade [pseudoprimo]
senão b = b^2 mod n
retorne false [composto]

Atividade 09

Crie um algoritmo baseado no MILLER-RABIN para obter números primos (ou pseudoprimos). Qual é a complexidade de tempo e espaço esperada para o algoritmo? Teste pelo menos 20 números primos e 20 não primos acima de 1000. Verifique com pelo menos 3 valores de s distintos. Quantos números primos e não primos foram corretamente identificados? Quais as vantagens de se utilizar MILLER-RABIN frente a outros métodos de obtenção de números primos como o crivo de eratóstenes?

Envie os exercícios para rafaelberri@gmail.com Assunto: "TC-CAL09"
Anexar: resultados/código fonte

Criptografia RSA (1)

Algoritmo:

- Selecionar dois números primos grandes $\boldsymbol{p}$ e $\boldsymbol{q}$, (512 bits, cada por exemplo) sendo $p \neq q$
- Calcular: $\boldsymbol{n = p * q}$
- Selecionar um inteiro ímpar "pequeno" $\boldsymbol{e}$ tal que $\boldsymbol{e}$ seja primo relativo de $(p - 1)(q - 1)$ [número primo]:

$$mdc\ (e, (p - 1)(q - 1)) = 1$$

Chave pública = (*e*, *n*)

Por quê (p-1)(q-1)?
https://crypto.stackexchange.com/questions/5715/phipq-p-1-q-1

Divisores Comuns

Um número é dito divisor comum se ele divide dois números:

$d \mid a$ e $d \mid b$ => d é divisor comum de a e b

Propriedade dos divisores comuns:

$d \mid a$ e $d \mid b$ implica em $d \mid (ax + by)$

O máximo divisor comum entre dois números
a e b é denotado por: $mdc\,(a\,,\,b)$

$d \mid a$ e $d \mid b$ então $d \mid mdc\,(a\,,\,b)$

Divisores Comuns

Primos relativos ou **Primos entre si** ou **Co-Primos:**

Dois inteiros são chamados de primos relativos se o único inteiro positivo que divide os dois é 1: $mdc\,(a\,,\,b) = 1$.

Por exemplo, 49 e 15 são primos relativos:

49 → 1,7,49

15 → 1,3,5,15

Propriedade

se $mdc\,(a\,,\,p) = 1$ e $mdc\,(b\,,\,p) = 1$ então $mdc\,(ab\,,\,p) = 1$

Testar: a=49, b=15, p=13

Fatoração

Fatoração Única

Um inteiro pode ser escrito como um produto da forma:

$$a = {p_1}^{e_1} \times {p_2}^{e_2} \times \ldots \times {p_r}^{e_r}$$

Exemplo

$$6.000 \quad = \quad 2^4 \times 3^1 \times 5^3 \quad = \quad 16 \times 3 \times 125$$

Teste de primalidade: Dado um número *n*, determinar se *n* é primo ("fácil")

Fatoração de inteiros: Dado um número *n*, representar *n* através de seus fatores primos (difícil – até o momento)

Divisores Comuns

Algoritmo de Euclides (MDC)

EUCLID (a , b)

se b = 0

então retorne a

senão retorne EUCLID(*b* , *a mod b*)

Calcular:

EUCLID(2 , 0)

EUCLID(99 ,78)

Complexidade:

Números pequenos: O (log b)

Números grandes (k bits): O (k^2) (**mod*)

Divisores Comuns

Algoritmo de Euclides Extendido

Adaptar o algoritmo anterior para calcular x e y em:

$$d = mdc\,(a,b) = ax + by$$

https://planetcalc.com/3298/

Calcular:

EXT-EUCLID(4 , 0)

EXT-EUCLID(99 ,78)

EXT-EUCLID (a , b)

se b = 0

então retorne (a , 1 , 0)

(d’, x’, y’) = EXT-EUCLID(*b* , *a mod b*)

(d , x , y) = (d’, y’, x’ – a / b * y’)

retorne (d , x , y)

Divisão inteira

Criptografia RSA (2)

Algoritmo:

- Calcular ***d*** como o inverso modular de ***e***:

$$e*d \equiv 1 \bmod ((p-1)(q-1))$$

Chave privada = (*d*, *n*)

Aritmética Modular

Soluções para a equação $ax \equiv b \pmod{n}$

Só há solução se: $mdc(a, n) \mid b$

$ax \equiv b \ (mod\ n)$ tem $\boldsymbol{d}$ soluções distintas

onde $d = mdc(a, n)$

MOD-LIN-SOLVER(a , b , n)

(d, x', y') = EXT-EUCLID(a, n)

se $d \mid b$

então $x_0 = x'(b / d) \bmod n$

para $i=0$ *a* d - 1 faça

imprimir($x_0 + i(n / d)) \bmod n$

senão imprimir "nenhuma solução"

https://planetcalc.com/3311/

Calcular:

14x ≡ 30 (mod 100)

MOD-LIN-SOLVER (14 , 30 , 100)

Aritmética Modular

Inverso multiplicativo modular:

O inverso multiplicativo modular de um inteiro ***a*** no módulo ***m*** é um inteiro ***x*** tal que:

$$ax \equiv 1\ (mod\ m)$$

* Existe se e somente se ***a*** e ***m*** são primos relativos.

$$17x \equiv 1 \text{ (mod 120)}$$

MOD-LIN-SOLVER (17 , 1 , 120)

Criptografia RSA

Transforma um inteiro M (que representa um bloco de dados da mensagem) em um inteiro C (que representa um bloco da mensagem criptografada), usando a seguinte função:

$$C = M^e \bmod n$$

Utilizar exponenciação modular!

Criptografia RSA

A transformação da mensagem criptografada C na mensagem original é executada através da formula:

$$M = C^d \bmod n$$

Utilizar exponenciação modular!

Cuidado!!

n deve ser maior do que M!

Caso contrário existem múltiplas interpretações para a mensagem codificada.
Se M for maior do que n, deve-se dividir a mensagem em blocos

Criptografia RSA (Exemplo)

Mensagem: "ola" => 111 105 97 => 01101111 01101001 01100001

A princípio nossa mensagem M teria o valor decimal: 7.301.473

Vamos utilizar **p = 521** e **q = 383**,

logo n = p*q = 199.543 e (p-1) * (q-1) = 198.640

Como M > n, devemos dividir a mensagem M em blocos.

Vamos usar blocos de dois caracteres!

Escolher arbitrariamente um valor para *e* [primo relativo a (p-1)(q-1)]

e = 227 [primo "pequeno"]

note que: mdc (227, 198.640) = 1 ***Chave pública* = (227, 199.543)**

Criptografia RSA (Exemplo)

p = 521 e **q = 383** | n = 199.543 | e = 227

Para gerar a chave privada precisamos calcular ***d*** como o inverso modular de ***e***:

$$e*d \equiv 1 \pmod{(p-1)(q-1)} \quad = \quad 227 * d \equiv 1 \pmod{198.640}$$

Usamos Euclides Estendido (227, 198.640) => mdc =1 / x =-*92.757* / y = *106*

E assim a única solução válida para a equação modular é:

$d = x\ (1 / 1) \bmod n = -92.757 \ \bmod\ 198.640 = 105.883$

***Chave privada* = (105.883, 199.543)**.

Mensagem M: 01101111 01101001 01100001 00000000

$M_1 = 57.193$ / $M_2 = 24832$

Codificando	Decodificando
$C_1 = 57.193^{227} \bmod 199.543 = 34.997$	$M_1 = 34.997^{105883} \bmod 199.543 = 57.193$
$C_2 = 24.832^{227} \bmod 199.543 = 61.019$	$M_2 = 61.019^{105883} \bmod 199.543 = 24.832$

Ataque Força Bruta ao RSA

Considerando que as mensagens criptografadas forem capturadas por terceiros, um ataque de **força bruta** é um ataque em que testa-se uma a uma todas as combinações possíveis para se quebrar (descobrir) uma chave privada.

No caso do RSA o “atacante” irá usar o valor n da chave pública para fatorar os valores de p e q.

Uma vez descoberto p e q basta calcular a chave privada e transformar a mensagem criptografada.

Qual a complexidade do algoritmo Força Bruta criado?

Envie os exercícios para rafaelberri@gmail.com Assunto: “TC-CAL10”
Anexar: resultados/código fonte