Criptografia - Intro

Ivan Sendin

FACOM - Universidade Federal de Uberlândia
ivansendin@yahoo.com,sendin@ufu.br

6 de junho de 2024

Exponencial e Polinomial

- Polinomial: factível, tratável
- Exponencial: intratável
- 2^{160} esta além do poder computacional **mundial**
- NP é fácil verificar
- certificado/informação/solução
- máquina não determinística
- Quase **todos** os problemas estudados são NP
- Verificar se um numero é par, verificar se um grafo é um ciclo
- Verificar se um grafo é bipartido
- **Achar** a solução pode ser fácil ou difícil

- Grande parte da criptografia só funciona com números grandes
- (só é segura)
- Exponencial X Polinomial
- Fatoração X Multiplicação
- *trapdoor*
- CPU: 64 bits
- (lebrando que os numeros grandes/nativos são ponto flutuante....)

- Biblioteca de Números Grandes
- Limite: memoria e CPU
- Os numeros são armazenados como vetores...
- As bibliotecas implementam as operações usuais: $+$,-,*,$\div$, ehPrimo,...

Exponenciação


```
int potencia(b,e):

    return  //b elevado a e
```

Como implementar ?

Exponenciação

Solução ingênua:

```
int potencia(b,e):
    return  r = b*b*b*b...
```

Exponencial no tamanho de e!!!

Exponenciação


```
int potencia(b,e):
    //Divisao e conquista
    //caso base...
    Se o expoente é par
        temp = potencia(b,e/2)
        retorna temp*temp
    temp = potencia(b,e/2)
    retorna temp*temp*b
```

Ganho exponencial

O Problema do Logaritmo

- Dados x, b com

$$x = b^e$$

determinar e

- Fácil?? Compare com o cálculo da potencia Caro

O Problema do Logaritmo

Um pouco mais trabalhoso: busca exponencial
$b, b^2, b^4, b^8 \ldots$

```
% Encontra o maior e, no formato de potencia de 2,  tal que b^e < l
def _logExp(b,l):

    step =b
    oldstep=step
    c=1
    oldc=c
    while step<l:
        oldstep=step
        oldc=c
        c=c*2
        step=step*step
    return (oldc,oldstep)


def logExp(b,l):
    r=0
    while (l>1):
        temp = _logExp(b,l)
        l = l/temp[1]
        r+=temp[0]
    return r
```

Exemplo para b^{13}

O Problema do Logaritmo

- Enquanto for menor...
- Passos exponencialmente crescentes
- **Busca Exponencial**
- Preciso do comparador $<$
- Exponenciação e logaritmo tem a mesma "classe" de complexidade

- A criptografia gosta de numeros circulares
- $50 + 15 \equiv 5 \pmod{60}$
- $5^2 \equiv 2 \pmod{23}$
- (Os numeros diminuiram no mundo circular...)

O Problema do Logaritmo Discreto

- Dados x, b e n com

$$x = b^e \pmod{n}$$

determinar e

- $\pmod{n}$ faz o x ser "circular"
- n precisa ser escolhido adequadamente... Vou omitir a parte matematica e vcs precisam confiar que é um número **circular**
- Em um número circular

$$b^x \pmod{n} < b^{x+1} \pmod{n}?$$

O Problema do Logaritmo Discreto

- Eu consido calcular a exponenciação modular usando o algoritmo D&C
- Eu não consigo fazer a busca exponencial no mundo modular
  Sem $<$
- Exponenciação modular é *fácil*....
- Logaritmo Discreto é *difícil*....

Escolha de chaves

- Alice e Bob querem conversar de forma privada
- Eva (eavesdropper) fica escutando tudo...
- Alice e Bob conhecem um bom cifrador
- mas não compartilham nenhum segredo....
- Todos envolvidos conhecem o basico de matematica: multiplicação (exponenciação)
- Este é um cenário **razoável** na Internet

Escolha de chaves

- Alice, Bob e Eva conhecem uma base g
- (Alice pode escolher um numero aleatorio e colocar no blog dela...)
- Alice escolhe uma chave a e calcula g^a
- Bob escolhe uma chave b e calculae g^b
- Alice faz um bradcast de g^a, Bob faz de g^b
- Alice conhece: g, a, g^a, g^b
- Bob conhece: g, b, g^b, g^a
- Eva conhece: g, g^b, g^a

Escolha de chaves

- Alice conhece: g, a, g^a, g^b
- Alice faz $(g^b)^a$, obtendo g^{ab}
- Bob conhece: g, b, g^b, g^a
- Bob faz $(g^a)^b$, obtendo $g^{ba} = g^{ab}$
- Eva conhece: g, g^b, g^a
- Eva faz o que??
- g^{a+b}!!!
- A chave é g^{ab}

Troca de Chaves

- Ou *key agreement*
- Na versão discreta (PLD)
- Conhecido com Protocolo Diffie-Hellman (DH)
- Ralph Merkle[1]
  - 1974 (Graduação em UC Berkley)
  - Projeto rejeitado pelo professor
  - Artigo rejeitado pela ACM

  I am sorry to have to inform you that the paper is not in the main stream of present cryptography thinking and I would not recommend that it be published in the Communications of the ACM.

[1] https://en.wikipedia.org/wiki/Stigler's_law_of_eponymy