# Segurança em Sistemas

## Aula 3 – Algoritmos e Aplicações de Segurança.

Prof. Filipo Mór
www.filipomor.com

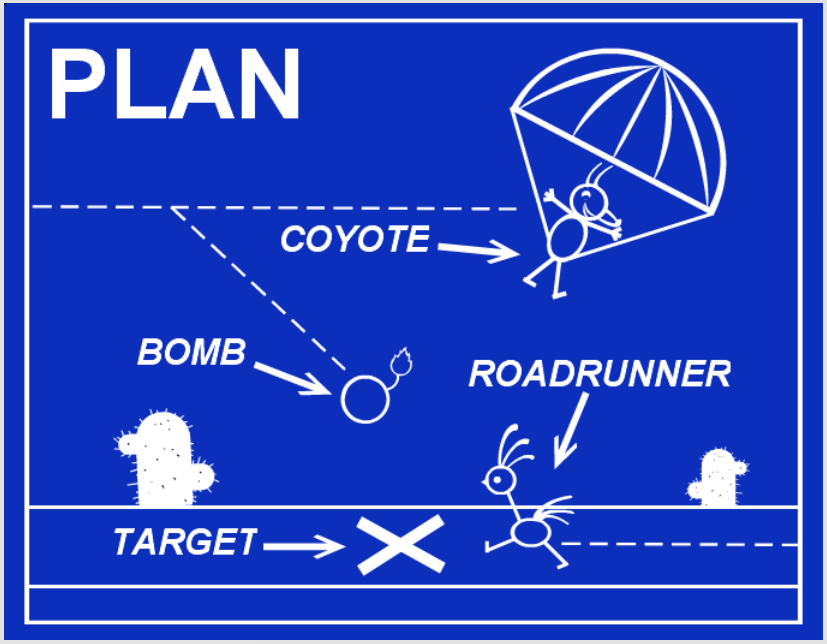
# Agenda

- Conceitos Básicos.
- Algoritmos para Criptografia.
- Algoritmos de Checksum.
- Algoritmos de Pesquisa em String.

# Objetivos

Objetivos da aula de hoje:

- ✓ Apresentar os conceitos básicos e algoritmos que formarão a base de estudo da área de criptografia e segurança de redes e sistemas.
- ✓ Propiciar o contato do estudante com algoritmos avançados utilizados na segurança de redes e sistemas.

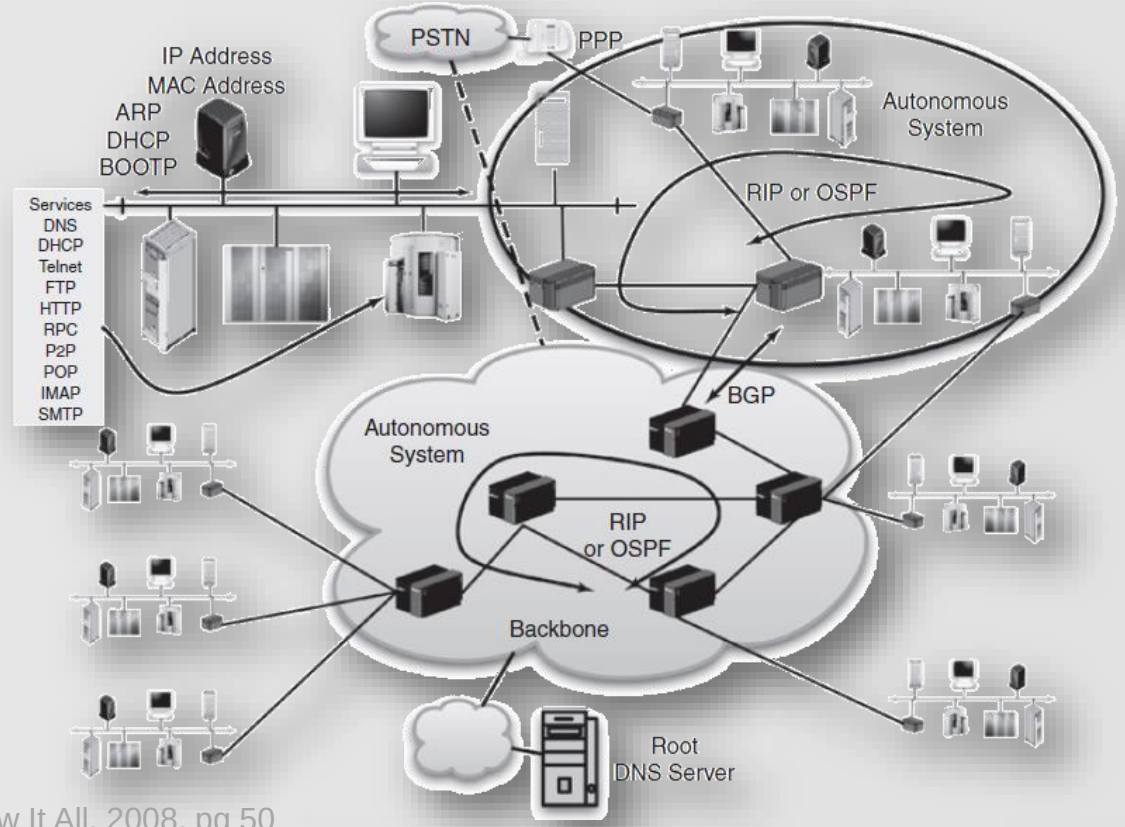


# Conceitos Básicos

- Complexidade da Comunicação em Rede
  - Grande quantidade de padrões e protocolos envolvidos, com diversas implementações em hardware e software.
  - A padronização permite a comunicação entre sistemas e aplicações de diferentes naturezas, como por exemplo entre máquinas com diferentes sistemas operacionais.
  - No entanto, tal diversidade acaba facilitando a criação de brechas na segurança.

# Conceitos Básicos

- Complexidade da Comunicação em Rede



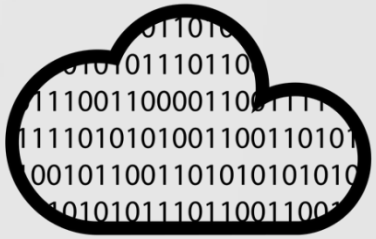

Network Security - Know It All, 2008, pg 50

# Conceitos Básicos

- Os problemas podem ser resultantes de:
  - Ataques.
  - Falhas.
  - Situações não previstas.
- Desta forma, são implementados métodos de conferência, validação e segurança em diversos níveis de abstração na comunicação em redes.

# Conceitos Básicos

- Algoritmos comumente utilizados:
  - Criptografia.
  - Validação de dados (checksum).
  - Busca de padrões em dados do tipo string.



# Algoritmos Básicos para Criptografia de Dados.

"What affected me most profoundly was the realization that the sciences of cryptography and mathematics are very elegant, pure sciences. I found that the ends for which these pure sciences are used are less elegant."

James Sanborn

# Algoritmos para Criptografia

- Criptografia
  - Do grego *kryptós*, "escondido", e *gráphein*, "escrita“.
  - Compreende na modificação do conteúdo da mensagem de forma que apenas o destinatário, em posse de uma chave de decriptografia, será capaz de acessar o seu conteúdo original.

# Algoritmos para Criptografia

- Relembrando a multiplicação de matrizes!

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} * \begin{vmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{vmatrix} \Rightarrow \begin{vmatrix} a_{11} * b_{11} + a_{12} * b_{21} & a_{11} * b_{12} + a_{12} + b_{22} \\ a_{21} * b_{11} + a_{22} * b_{21} & a_{21} * b_{12} + a_{22} * b_{22} \\ a_{31} * b_{11} + a_{32} * b_{21} & a_{31} * b_{12} + a_{32} * b_{22} \end{vmatrix}$$

***Exemplo 1***

$$\begin{vmatrix} 2 & 5 & 9 \\ 3 & 6 & 8 \end{vmatrix} * \begin{vmatrix} 2 & 7 \\ 4 & 3 \\ 5 & 2 \end{vmatrix} \Rightarrow \begin{vmatrix} (2*2)+(5*4)+(9*5) & (2*7)+(5*3)*(9*2) \\ (3*2)+(6*4)+8*5) & (3*7)+(6*3)+(8*2) \end{vmatrix}$$

$$\begin{vmatrix} 4+20+45 & 14+15+18 \\ 6+24+40 & 21+18+16 \end{vmatrix} \Rightarrow \begin{vmatrix} 69 & 47 \\ 70 & 55 \end{vmatrix}$$

Site Brasil Escola, 05/03/2015 - http://www.brasilescola.com/matematica/multiplicacao-matrizes.htm

# Algoritmos para Criptografia

- Metodo Básico de Criptografia.
  - Utilização de matrizes inversas: A e B = $A^{-1}$
  - Exemplo:

$$A = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix} \quad \text{e} \quad B = \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix}$$

Para converter a mensagem para a forma numérica utilizaremos a seguinte tabela de conversão:

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | . | , | b̲ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 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 |

Exemplo extraído do livro Introdução à ÁLGEBRA LINEAR.
C.H.Edwards, Jr.
David E. Penney

# Algoritmos para Criptografia

- A mensagem "*the game is afoot*" será convertida para o formato numérico da seguinte forma:

**the game is afoot**

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | . | , | b |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 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 |

| T | H | E | b | G | A | M | E | b | I | S | b | A | F | O | O | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | 8 | 5 | 29 | 7 | 1 | 13 | 5 | 29 | 9 | 19 | 29 | 1 | 6 | 15 | 15 | 20 |

# Algoritmos para Criptografia

- Como a matriz codificadora A é 2x2, então a mensagem convertida para formato numérico será organizada no seguinte padrão na matriz M:

| T | H | E | b | G | A | M | E | b | I | S | b | A | F | O | O | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | 8 | 5 | 29 | 7 | 1 | 13 | 5 | 29 | 9 | 19 | 29 | 1 | 6 | 15 | 15 | 20 |

$$M = \begin{bmatrix} 20 & 8 & 5 & 29 & 7 & 1 & 13 & 5 & 29 \\ 9 & 19 & 29 & 1 & 6 & 15 & 15 & 20 & 29 \end{bmatrix}$$

# Algoritmos para Criptografia

- Para codificação da mensagem, faz-se:

$$N = AM$$

$$N = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 20 & 8 & 5 & 29 & 7 & 1 & 13 & 5 & 29 \\ 9 & 19 & 29 & 1 & 6 & 15 & 15 & 20 & 29 \end{bmatrix}$$

$$N = \begin{bmatrix} 69 & 43 & 44 & 88 & 27 & 18 & 54 & 35 & 116 \\ 49 & 35 & 39 & 59 & 20 & 17 & 41 & 35 & 87 \end{bmatrix}$$

# Algoritmos para Criptografia

- Entao, a mensagem codificada resultante será a lista de números inteiros:

{ 69, 43, 44, 88, 27, 18, 54, 35, 116, 49, 35, 39, 59, 20, 17, 41, 30, 87 }

Que deverá ser enviada ao destinatário.

# Algoritmos para Criptografia

- Para decodificar a mensagem, o destinatário deverá efetuar:

$$M = BN$$

$$\mathrm{M}= \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} 69 & 43 & 44 & 88 & 27 & 18 & 54 & 35 & 116 \\ 49 & 35 & 39 & 59 & 20 & 17 & 41 & 35 & 87 \end{bmatrix}$$

$$M = \begin{bmatrix} 20 & 8 & 5 & 29 & 7 & 1 & 13 & 5 & 29 \\ 9 & 19 & 29 & 1 & 6 & 15 & 15 & 20 & 29 \end{bmatrix}$$

# Algoritmos para Criptografia

$$M = \begin{bmatrix} & & & & & & & & & & & & & & & & & & & & & & & & & & & 29 \\ & & & & & & & & & & & & & & & & & & & & & & & & & & & 29 \end{bmatrix}$$

| A | B | C | D | E | F | X | Y | Z | . | , | b |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 24 | 25 | 26 | 27 | 28 | 29 |

**the game is afoot**

# Algoritmos para Criptografia

- Analisando o algoritmo:

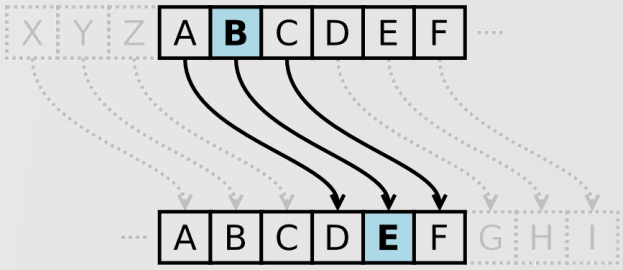
```
// Multiplica as matriz A e B
for(i=0; i<N; i++)
    for(j=0; j<N; j++)
        for(k=0; k<N; k++)
        {
            C[i*N+j] += A[i*N+k] * B[k*N+j];
        }
```

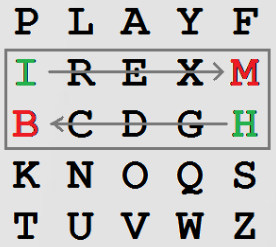
- Complexidade: $O(n^3)$
  - (considerando-se matrizes quadradas $NxN$).
- São necessárias duas chaves: as matrizes A e B.

# Algoritmos para Criptografia

- Outros métodos básicos de criptografia:
  - Aritmética modular.
  - Cifra de César.
  - Quadrado de Vinegère.
  - Cifra Monoalfabética.
  - Cifra Playfair.

$$a_1 + a_2 \equiv b_1 + b_2 \pmod{n}$$
$$a_1 - a_2 \equiv b_1 - b_2 \pmod{n}$$

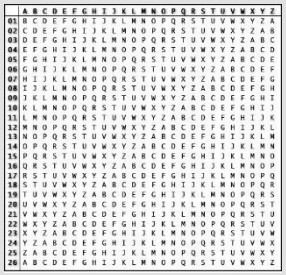



HI

Shape: Rectangle
Rule: Pick Same Rows, Opposite Corners

BM



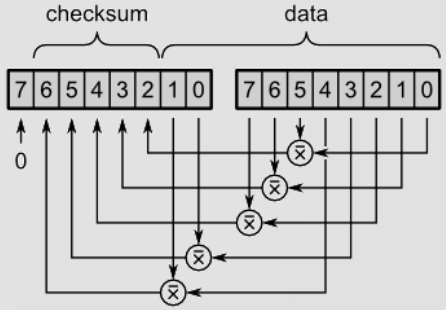
# Algoritmos para Verificação de Dados (Checksum)

Social security, bank account, and credit card numbers aren't just data. In the wrong hands they can wipe out someone's life savings, wreck their credit and cause financial ruin.

Melissa Bean

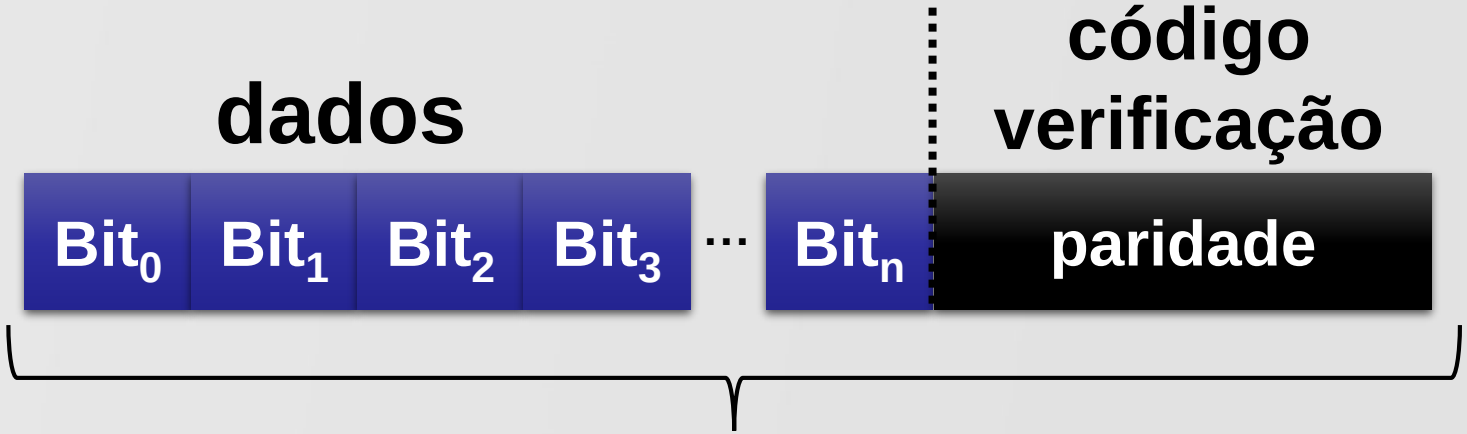
# Algoritmos de Checksum

- Checksum
  - Consiste na verificação do dado recebido pela comparação do código verificador recebido contra o código de verificação calculado.
  - Métodos mais utilizados:
    - Cálculo de Paridade por Byte ou Palavra.
    - Verificação Longitudinal por Redundância.
    - Redundância Cíclica.
    - Soma Modular.
    - Método de Fletcher.
    - Método de Adler.
    - Método de Luhn.



# Algoritmos de Checksum

- Cálculo de Checksum por Paridade:




$$paridade = Bit_0 \oplus Bit_1 \oplus Bit_2 \oplus Bit_3 \oplus \cdots \oplus Bit_{n-1}$$

XOR (OU exclusivo)

# Algoritmos de Checksum

- Cálculo de Checksum por Paridade:

$$\textbf{paridade} = \textbf{Bit}_0 \oplus \textbf{Bit}_1 \oplus \textbf{Bit}_2 \oplus \textbf{Bit}_3 \oplus \cdots \oplus \textbf{Bit}_{n-1}$$

**Tabela Verdade XOR**

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

Paridade para números de 1 bit:
Número Par: 0
Número Ímpar: 1

# Algoritmos de Checksum

- Algoritmo de Luhn:

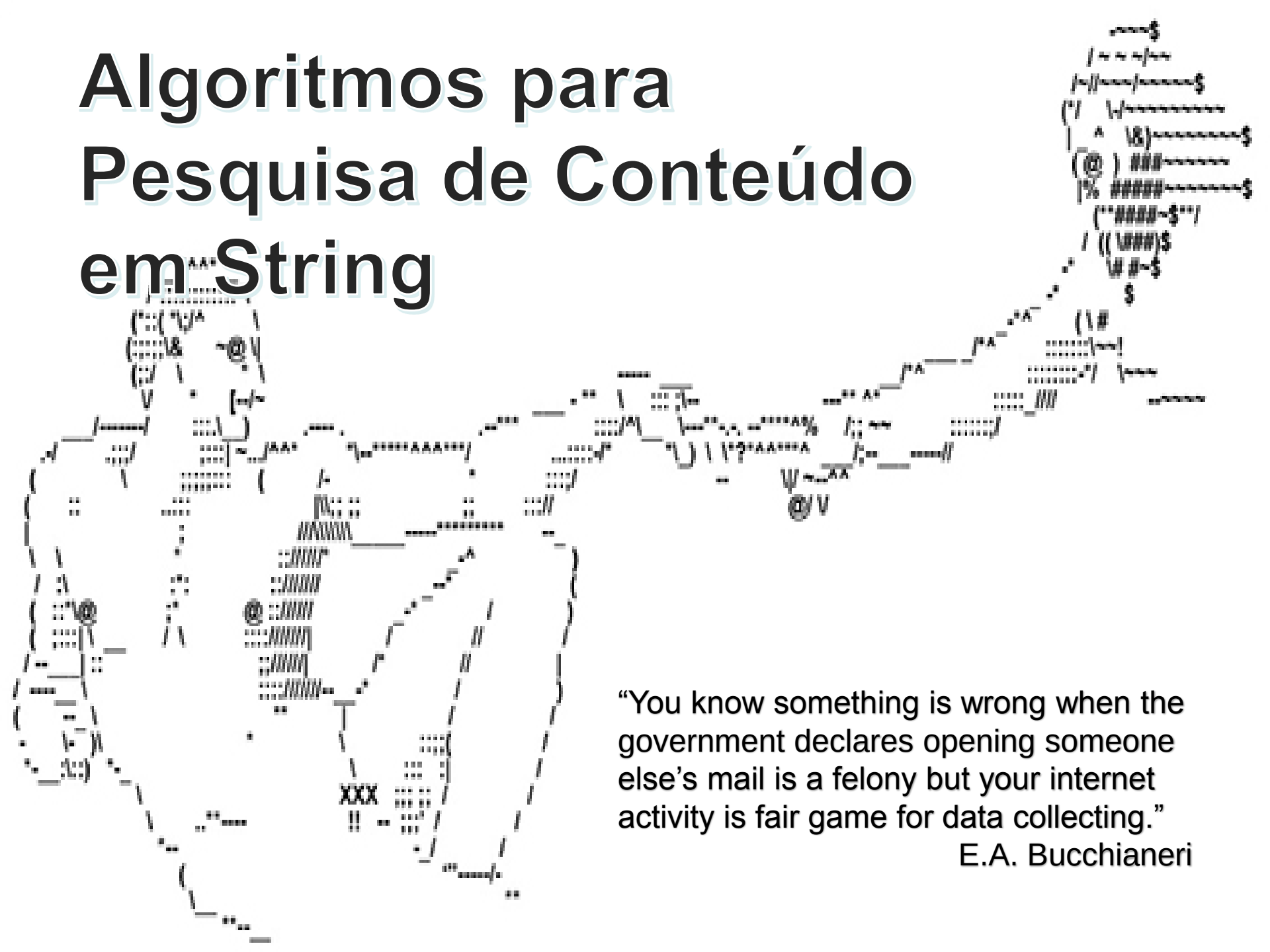
```
sum = 0
size = length(C)
parity = size mod 2
for i from 0 to size - 1
{
    digit = C[i]
    if (i mod 2 == parity)
        digit = 2 * digit

    if (digit > 9)
        digit = digit - 9

    sum += digit
}
return (sum mod 10) = 0
```

- Complexidade: $O(n)$
- Utilizado para validar:
  - Números de cartão de crédito.
  - Códigos IMEI (celulares).
  - Telefonia via satélite.
  - Números seguridade social nos EUA e Canadá.
  - Fraqueza: não detecta transposição de algarismos (por exemplo, “08” por “80”)

# Algoritmos para Pesquisa de Conteúdo em String



"You know something is wrong when the government declares opening someone else's mail is a felony but your internet activity is fair game for data collecting."

E.A. Bucchianeri

# Algoritmos de Pesquisa em String

- Métodos para Pesquisa de Conteúdo em String:
  - Verificação de pacotes recebidos via rede.
  - Detecção de *worms*.
  - Proteção de scripts.
  - Outras áreas do conhecimento:
    - Big data.
    - Estudo do genoma humano.
    - Data mining.

# Algoritmos de Pesquisa em String

- Algoritmos de busca mais utilizados:
  - Algoritmo de Boyer-Moor.
    - Considerado o mais eficiente algoritmo de busca em string.
    - A busca é sempre realizada da direita para a esquerda.
    - Ideal para situações onde o texto sendo procurado é muito menor do que o texto base, ou quando o padrão sendo procurado se repete muitas vezes no texto base.
    - Se utiliza de informações geradas na fase de pré-processamento para “pular” trechos de código durante a fase de processamento.
    - Complexidade $O(n/m)$ (melhor caso) na fase de processamento.
    - Utilizado em funções de Search e Replace de editores de texto.

# Algoritmos de Pesquisa em String

- Algoritmos de busca mais utilizados:
  - Algoritmo de Boyer-Moor.
    - Fase de pré-processamento.

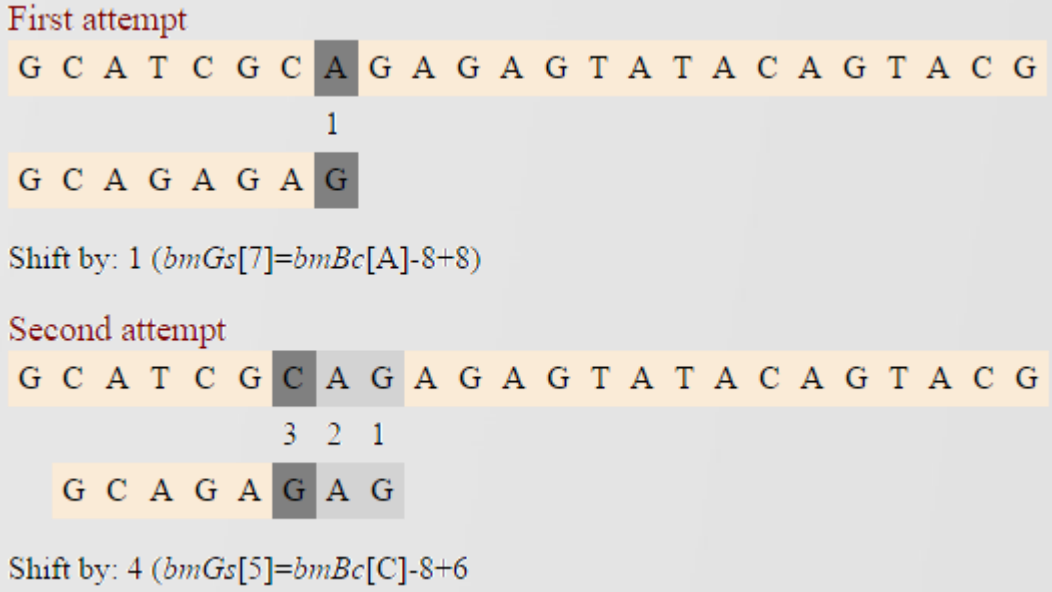
| $c$ | A | C | G | T |
|---|---|---|---|---|
| $bmBc[c]$ | 1 | 6 | 2 | 8 |

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $x[i]$ | G | C | A | G | A | G | A | G |
| $suff[i]$ | 1 | 0 | 0 | 2 | 0 | 4 | 0 | 8 |
| $bmGs[i]$ | 7 | 7 | 7 | 2 | 7 | 4 | 7 | 1 |

Exemplo extraído do site Boyer-Moore algorithm, http://www-igm.univ-mlv.fr/~lecroq/string/node14.html, 04/02/2015.

# Algoritmos de Pesquisa em String

- Algoritmos de busca mais utilizados:
  - Algoritmo de Boyer-Moor.
    - Fase de processamento (busca).




Exemplo extraído do site Boyer-Moore algorithm, http://www-igm.univ-mlv.fr/~lecroq/string/node14.html, 04/02/2015.

# Algoritmos de Pesquisa em String

Third attempt

```
G C A T C G C A G A G A G T A T A C A G T A C G
          8 7 6 5 4 3 2 1
          G C A G A G A G
```

Shift by: 7 (*bmGs*[0])

Fourth attempt

```
G C A T C G C A G A G A G T A T A C A G T A C G
                                  3 2 1
                G C A G A G A G
```

Shift by: 4 (*bmGs*[5]=*bmBc*[C]-8+6)

Fifth attempt

```
G C A T C G C A G A G A G T A T A C A G T A C G
                                        2 1
                      G C A G A G A G
```

Shift by: 7 (*bmGs*[6])

Exemplo extraído do site Boyer-Moore algorithm, http://www-igm.univ-mlv.fr/~lecroq/string/node14.html, 04/02/2015.

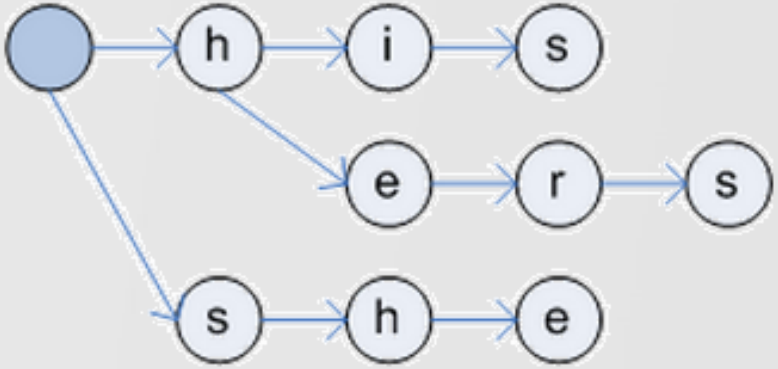
# Algoritmos de Pesquisa em String

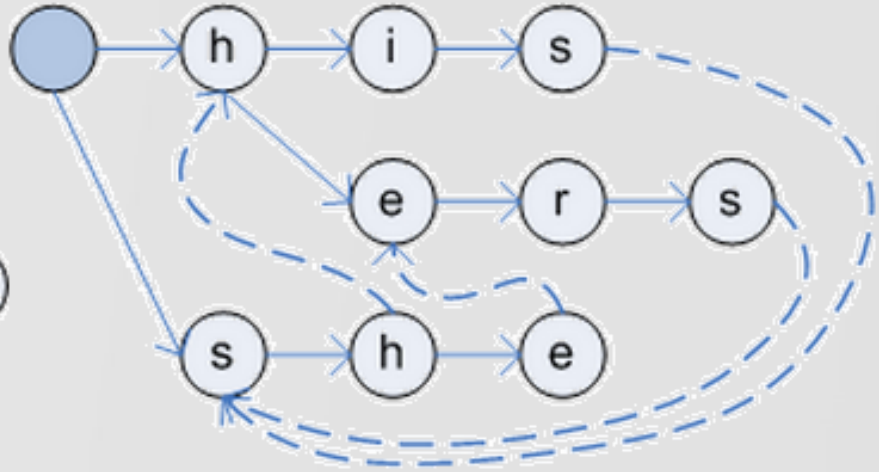
- Algoritmos de busca mais utilizados:
  - Algoritmo de Aho–Corasick.
    - Executa em $O(n + m + z)$, sendo $n$ o tamanho do texto base, $m$ o tamanho do trecho sendo procurado e $z$ a quantidade de ocorrências do trecho procurado no texto base.
    - Baseado na implementação de uma árvore de chaves de pesquisa.
    - A construção da árvore toma $O(n)$.
    - A pesquisa toma $O(m + z)$.
    - No melhor caso, pode executar em $\Theta(nm)$.

# Algoritmos de Pesquisa em String

$$P = \{he, she, his, hers\}$$

Simulação em Video (Youtube):
http://youtu.be/d24CyiU1JFk




Site Code Project, Aho-Corasick string matching in C#, 01/02/2015, http://www.codeproject.com/Articles/12383/Aho-Corasick-string-matching-in-C

# Algoritmos de Pesquisa em String

- Atividade:
  - Defina um algoritmo para pesquisa de trechos de texto dentro de um texto maior. Pode ser em português estruturado ou qualquer linguagem, desde que sejam utilizados comandos primitivos (sem o uso de bibliotecas de alto nível para pesquisa em string).
- Questões de Pesquisa:
  - Qual a complexidade (assimptotica) do seu algoritmo? Como ele reage alterando-se drasticamente o tamanho do texto procurado ou do texto base?
  - Funciona com textos de qualquer tamanho?
  - Detecta trechos de texto ligeiramente diferentes do texto procurado (por exemplo, detecta “*paerl.exe*” se voce procurer por “*pearl.exe*”)?

# Dúvidas e Comentários?

***Agradecimentos:***
Prof. Marcelo Conterato
Prof. Samuel Souza

# Segurança em Sistemas

**Faculdade SENAC de Porto Alegre**
Prof. Filipo Mór
www.filipomor.com

2017/II