



Pretty Good Privacy?

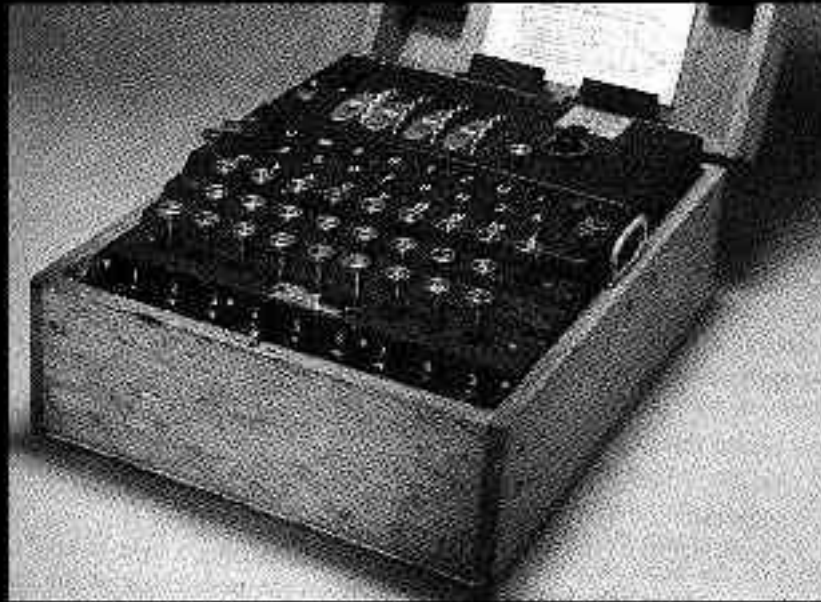
**APRESENTA**



<http://nerv.5p.org.uk/>

# CRYPTO & CLUSTER

( PARTE 2 )



**Objetivo: demonstrar técnicas de força bruta sobre criptografia difundida e discutir melhorias.**

# Tópicos

---

- ❑ Força bruta (outra vez?)
- ❑ Possíveis Avanços
- ❑ Em ambientes distribuídos
  - /etc/shadow
  - crypt(3)
  - pgp

# BF... Outra vez?

---

Esmagar, dilacerar, destruir...

Força Bruta

# Classificação

---

- BURRA

Baseada em dicionários...

- ESTÚPIDA

Tenta TUDO: aaaa, aaab, aaac...

Just kidding!

Brute Force

## Senhas inteligentes?

---

Várias ferramentas implementam vários testes antes de validar uma senha...

### Exemplo

**Login:** nibble

**Senhas:** nibble, nibblenibble, blenib, elbbin, nibble123,  
qwerasdf, nib666, ....

**Não são seriam aceitas!**

Brute Force

## Senhas inteligentes...

---

Estas visam forçar o usuário a seguir uma bastante conhecida recomendação.

Use uma senha “pessoal”.



# Constatações sobre frases senhas

---

- Pessoas tendem confiar mais nas frases senhas, usando assim menos caracteres especiais como “#a!ç£@→² \$m”.
- Frases senhas tendem a seguir uma construção normal de sentença com adicionados alguns elementos pessoais.

## Recomendações:

<http://www.stack.nl/~galactus/remailers/passphrase-faq.html>

<http://world.std.com/~reinhold/dicewarefaq.html>

# Possíveis Avanços

---

Keep walking

Força Bruta

## Reduzindo Universo

---

- Ataques contra textos em idioma conhecido podem ser otimizados...
- Ataques direcionados, onde textos são usados na geração listas de senhas...

Força Bruta

## Construindo possibilidades!

---

As línguas europeias possuem uma média de **300.000** palavras, onde:

- **140** palavras estruturais que representam cerca de 50% de um texto médio
- **+2.000** palavras temos o vocabulário básico, cerca de 85% de um texto médio
- **+2.500** palavras temos o vocabulário médio, cerca de 95% de um texto médio

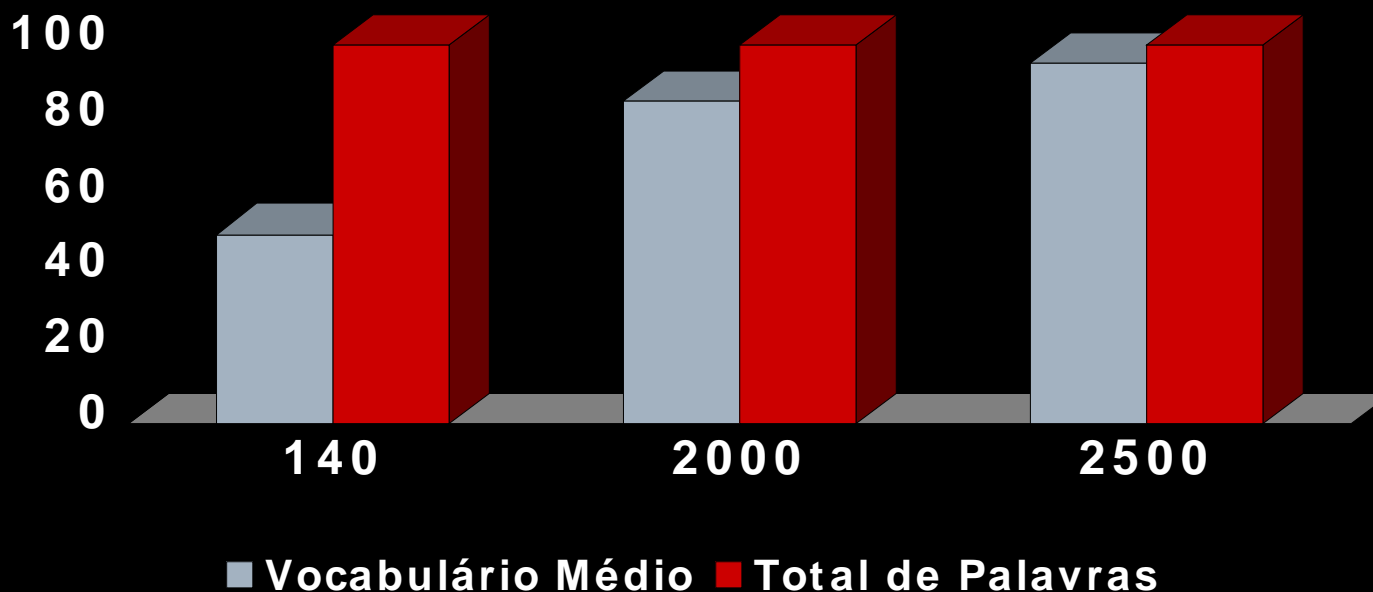
Logo...

Força Bruta

# Construindo possibilidades!

---

## Número de Palavras x Porcentagem em Texto



Força Bruta

# Construindo possibilidades!

---

Com esta lista de 2500 palavras mais alguns textos para capturar palavras pessoais obtém-se então uma base de dados excepcional...

...concordam? Respondam!

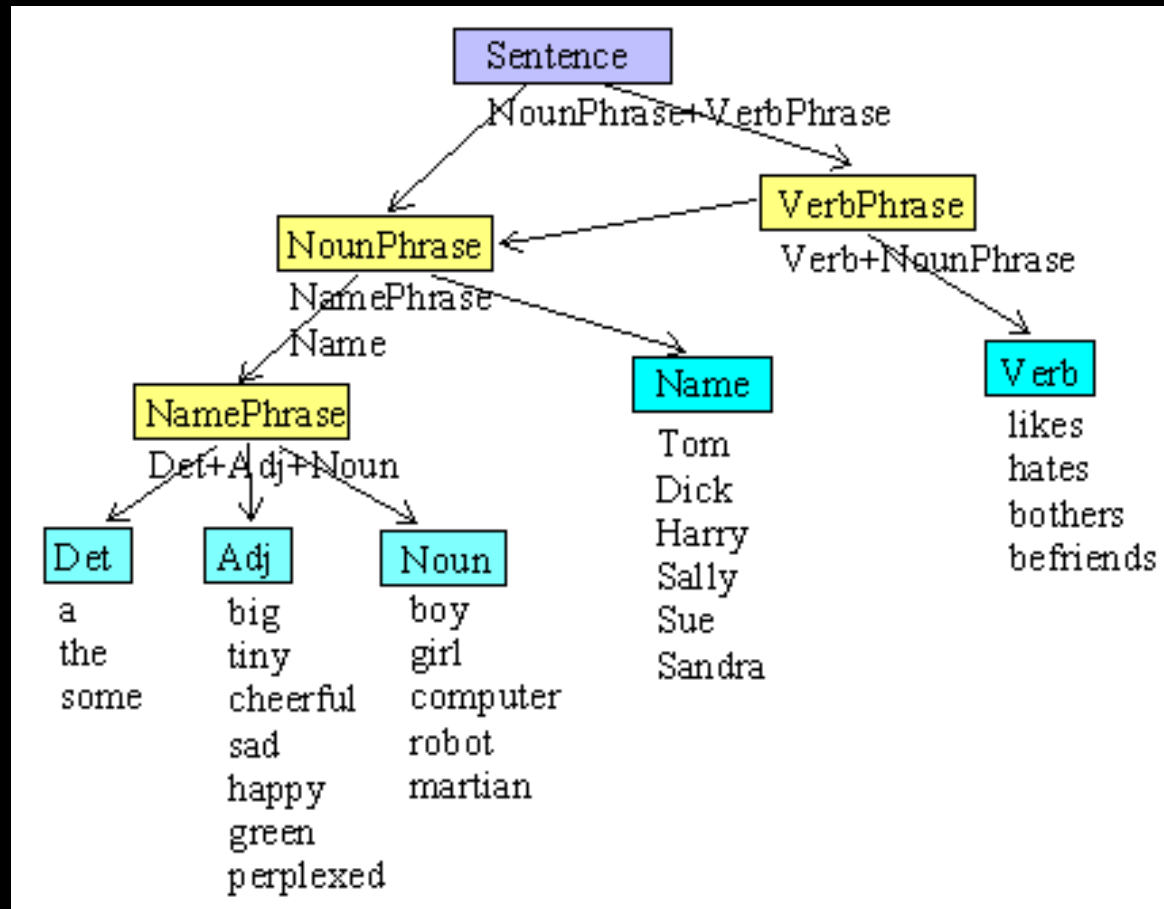
# E frases? É possível gerar?

---

Ferramentas (tal como o pgp) instigam usuários às “frases senhas” ...

Surge um novo problema:  
**como gerar frases?**

# Complexidade





h2hc\_axon.avi

0' 33''

Start ID:      Capitalize first char    Next >>

Repeat:  times     No repeated words    Close

Trace steps

A cheerful computer hates Dick  
 Sally hates Sally  
 The happy boy bothers the sad boy  
 Harry hates the big boy  
 A sad robot bothers Sue  
 Some green girl bothers the perplexed boy  
 Sandra hates Sandra  
 The happy martian befriends Sandra  
 A big robot bothers a tiny boy



Tom  
Dick  
Harry  
Sally  
Sue  
Sandra



# Algoritmos

---

O algoritmo de markov é uma possível saída para gerar frases *compreensíveis*.

<http://cm.bell-labs.com/cm/cs/tpop/code.html>  
(Implementações em C, C++, Java, Perl...)

Força Bruta

## Filtrar palavras pessoais...

---

- ❑ Buscar as que mais se repetem, porém não estão entre as 140...
- ❑ Conhecida áreas de interesse, buscar nomes baseado (ainda) em listas...
- ❑ Dentre outras mil possibilidades...

/etc/shadow

---

Unix rules

Força Bruta

# Shadow Passwords

---

- Usado em vários unix(es) para proteger senhas de usuários.
- Faz uso de DES com crypt(3).

<http://www.tldp.org/HOWTO/Shadow-Password-HOWTO.html>

Força Bruta

# Quebrando Shadow

---

□ John the Ripper, by Solar Designer

<http://www.openwall.com/john/>

□ Crack, by Alec Muffett

<http://www.crypticide.com/users/alecm/>

Força Bruta

10 pcs pentium 200 Mhz

# of characters	26 lower case	36 lower & numeric	62 alpha & numeric	95 printable	128 ASCII
5	92s	7.8m	118m	16.6h	73.5h
6	40m	4.7h	121.4h	65.6d	391.6d
7	17.2h	167.5h	313.6d	17.1y	137.3y
8	18.6d	251.2d	53.3y	1618y	175.7c

<http://personal.stevens.edu/~khockenb/crypt3.html>



Força Bruta

# John The Ripper

---

“ Its primary purpose is to detect weak  
Unix passwords.”

- Solar Designer

<http://www.openwall.com/john/>

h2hc\_john.avi

0' 34''

```
nibble@pamela: ~/app/djohn/john/run$ rm john.pot
rm: cannot remove 'john.pot': No such file or directory
nibble@pamela: ~/app/djohn/john/run$ cat passwd.1
root:$1$/gk//f6hSkEhRtdhdbBkLvbEjJwqBUX1:12934:0:99999:7:::
nibble:81$&dr+4TbOXQIaENTSEFhRdcXHmLRHQH /:12934:0:99999:7:::
fart:$1$Jy212nAKSfbc.7YicwvEjEUMHleHzj.:13076:0:99999:7:::
nert:$1$HrK1pM8KSHOKhr6TQ1mTMabb/cT1O/:13076:0:99999:7:::
hert:$1$GHF000UK$nvqLX3dNOpCZi34T0TU210:13076:0:99999:7:::
nibble@pamela: ~/app/djohn/john/run$ john passwd.1
Loaded 5 passwords with 5 different salts (FreeBSD MD5 [32/32])
h3rt          (hert)
n3rt          (nert)
guesses: 2   time: 0:00:00:07 51% (1)  c/s: 572  trying: 9999910
Session aborted
nibble@pamela: ~/app/djohn/john/run$ cat john.pot
$1$Gm0008K$nvqLX3dNOpCZi34T018250:h3rt
$1$HrK1pM8KSHOKhr6TQ1mTMabb/cT1O/:n3rt
nibble@pamela: ~/app/djohn/john/run$ john passwd.1
Loaded 3 passwords with 3 different salts (FreeBSD MD5 [32/32])
```

Força Bruta

## Distributed John (djohn)

---

“With DJohn you can crack passwords using several machines to get passwords sooner than using a single machine.”

- Luis Parravicini

<http://ktulu.com.ar/en/djohn.php>

h2hc-djohn.avi

1' 49''

```
ALICIA
nibble@alicia:~/app/djohn$ cat passwd.1
cert:$1$HrKlpMBK$HOAKhz6IQImTMabb/cJto/:13076:0:99999:7:::
nibble@alicia:~/app/djohn$ djohn pamela
guesses: 0 time: 0:00:00:02 c/s: 5000 trying: ai01
guesses: 0 time: 0:00:00:02 c/s: 5000 trying: appt
guesses: 0 time: 0:00:00:02 c/s: 5000 trying: axf1
guesses: 0 time: 0:00:00:02 c/s: 5000 trying: b55d
guesses: 0 time: 0:00:00:02 c/s: 5000 trying: bkkx
```

```
PAMELA
djohn -b 192.168.0.1 -n a-z0-9 -f $1 -l $2 -w 10000 -v
nibble@pamela:~/app/djohn$ djohn.sh aaaa 9999
Starting on Fri Oct 21 10:16:14 2005
using serial id 1129896974
Fri Oct 21 10:16:28 192.168.0.2:32768 GET_CHARACTER
Fri Oct 21 10:16:28 192.168.0.2:32768 GET_CODE
Fri Oct 21 10:16:28 192.168.0.2:32768 GET_WORK 1129897036 aaaa ai01 new 0
Fri Oct 21 10:16:32 192.168.0.2:32768 GET_WORK 1129897042 ai02 appt new 1
Fri Oct 21 10:16:36 192.168.0.2:32768 GET_WORK 1129897091 appu axf1 new 2
Fri Oct 21 10:16:41 192.168.0.2:32768 GET_WORK 1129897119 axfm b55d new 3
Fri Oct 21 10:16:43 192.168.0.2:32768 GET_CHARACTER
Fri Oct 21 10:16:43 192.168.0.2:32768 GET_CODE
Fri Oct 21 10:16:43 192.168.0.2:32768 GET_WORK 1129897169 b55e bcv5 new 4
Fri Oct 21 10:16:45 192.168.0.2:32768 GET_WORK 1129897191 bcv6 bkkx new 5
```

```
ASHLEY
nibble@ashley:~/app/djohn$ cat passwd.1
cert:$1$HrKlpMBK$HOAKhz6IQImTMabb/cJto/:13076:0:99999:7:::
nibble@ashley:~/app/djohn$ djohn pamela
guesses: 0 time: 0:00:00:03 c/s: 3333 trying: bcv5
```

Força Bruta

# Desvantagens do DJohn

---

- ❑ Todos os clientes recebem mesma a quantidade de trabalho...
- ❑ Arquivo de senhas deve ser passado manualmente para os clientes...

crypt(3)

---

Unix rules



Força Bruta

## crypt(3): implementação

---

```
#include <unistd.h>
```

```
char *crypt (char *key, char *salt);
```

5. *key* é a sentença a ser encriptada.
6. *salt* são 2 caracteres “pertubar” o algoritmo.

<http://www.opengroup.org/onlinepubs/007908799/xsh/crypt.html>

# bf-crypt

Lança 26 processos, “migraveis”  
sob openmosix, contra crypt()

```
1  /* b4.c */
2
3  #include <stdio.h>
4  #include <unistd.h>
5  #include <stdlib.h>
6
7  char *crypt(const char *key, const char *salt);
8  extern char *getpass();
9
10 void stripnl(char *str);
11
12 int main() {
13     //char alpha[62] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
14     //char alpha[36] = "abcdefghijklmnopqrstuvwxyz0123456789";
15     char alpha[26] = "abcdefghijklmnopqrstuvwxyz";
16
17     char buf[5], salt[3];
18     char *password, encpasswd[15];
19     int a, b, c, d;
20
21     fgets(encpasswd, sizeof(encpasswd), stdin);
22     stripnl(encpasswd);
23     salt[0] = encpasswd[0]; salt[1] = encpasswd[1]; salt[2] = 0;
24     memset(buf, 0, 5);
25     a = b = c = d = 0;
26
27     for (a = 0; a < sizeof(alpha); a++) { buf[0] = alpha[a];
28         if (!fork()) {
29             for (b = 0; b < sizeof(alpha); b++) { buf[1] = alpha[b];
30                 for (c = 0; c < sizeof(alpha); c++) { buf[2] = alpha[c];
31                     for (d = 0; d < sizeof(alpha); d++) { buf[3] = alpha[d];
32                         if(!strcmp(crypt(buf, salt), encpasswd)) {
33                             printf("\nPassword (%s) validated!\n", buf);
34                             system("./kill.sh b4");
35                         } } } } }
36     return 0;
37 }
38
39 void stripnl(char *str) {
40     while(strlen(str) && ( (str[strlen(str) - 1] == 13) ||
41         ( str[strlen(str) - 1] == 10 ))) {
42         str[strlen(str) - 1] = 0;
43     }
44 }
```

**Mais detalhes...**

# Definição do *charset*

```
13 //char alpha[62] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
14 //char alpha[36] = "abcdefghijklmnopqrstuvwxyz0123456789";
15 char alpha[26] = "abcdefghijklmnopqrstuvwxyz";
```

# Construindo “palavras”

```
27 for (a = 0; a < sizeof(alpha); a++) { buf[0] = alpha[a];
28     if (!fork()) { // atrelado ao charset
29         for (b = 0; b < sizeof(alpha); b++) { buf[1] = alpha[b];
30             for (c = 0; c < sizeof(alpha); c++) { buf[2] = alpha[c];
31                 for (d = 0; d < sizeof(alpha); d++) { buf[3] = alpha[d];
32                     if(!strcmp(crypt(buf, salt), encpasswd)) {
33                         printf("\nPassword (%s) validated!\n", buf);
34                         system("./kill.sh b4");
35                     } } } } } }
```

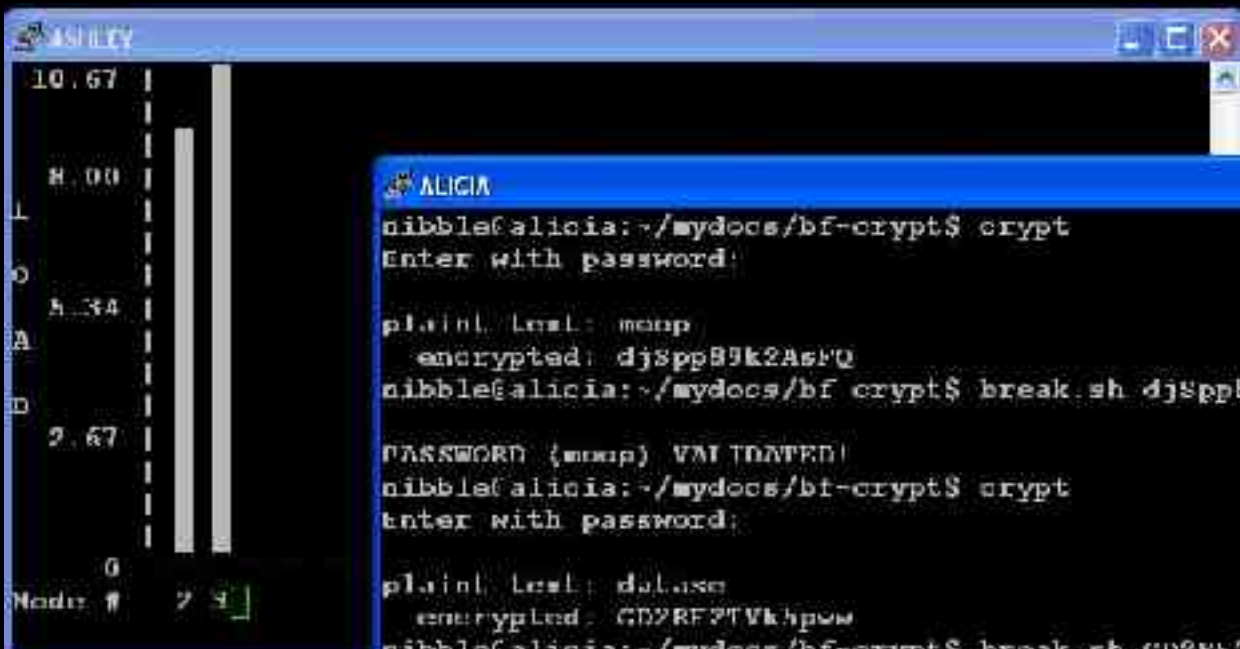
Força Bruta

## Por que usar OpenMosix?

---

- Programas cpu-bound não precisam de reprogramação e tendem a migrar...
- Prático possui ferramentas de gerenciamento e análise, tal como boot via cd via knoppix...

h2hc-crypto.avi



```

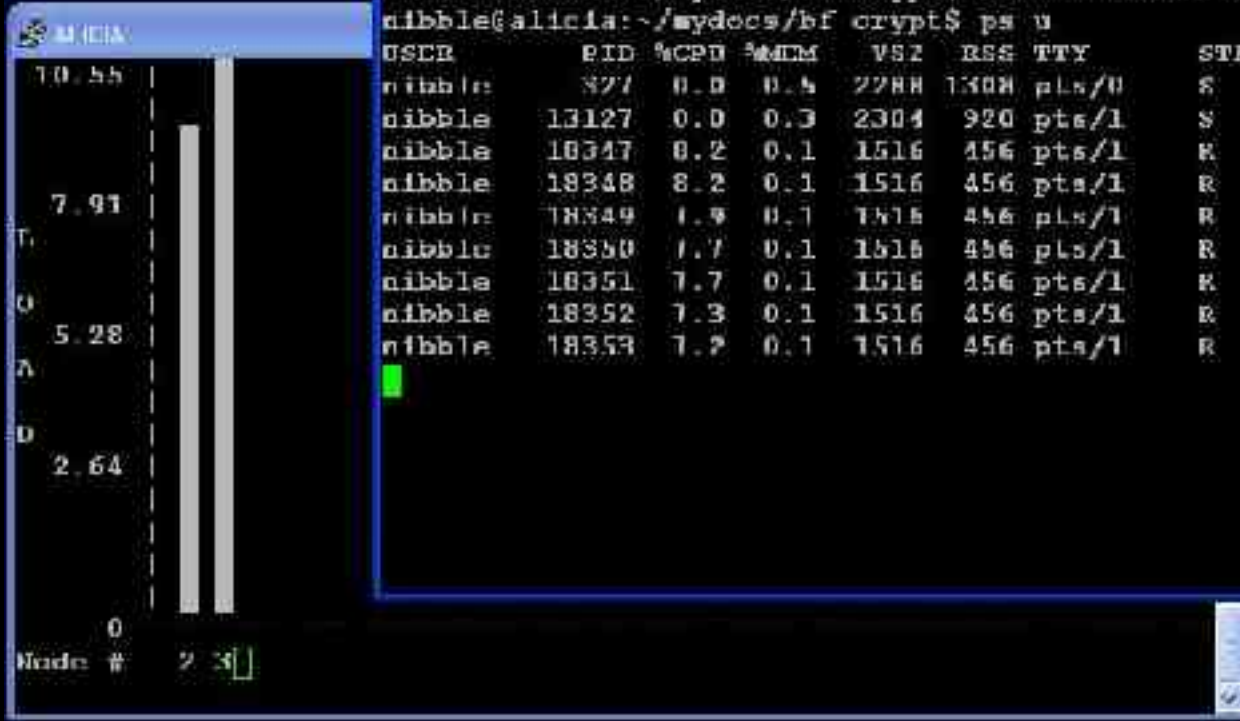
ALICIA
nibble@alicia:~/mydocs/bf-crypt$ crypt
Enter with password:

plain: test: mamp
  encrypted: djsppB9k2AsfU
nibble@alicia:~/mydocs/bf-crypt$ break.sh djsppB9k2AsfU 4

PASSWORD (mamp) VALIDATED!
nibble@alicia:~/mydocs/bf-crypt$ crypt
Enter with password:

plain: test: datase
  encrypted: GD2Kt21vk5pww
nibble@alicia:~/mydocs/bf-crypt$ break.sh GD2Kt21vk5pww 6
nibble@alicia:~/mydocs/bf-crypt$ ps u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
nibble:   827  0.0  0.5  2288  1308 pts/0    S    00:25   0:00 -bash
nibble:  13127  0.0  0.3  2304   920 pts/1    S    02:02   0:00 -bash
nibble:  18347  0.2  0.1  1516   456 pts/1    R    03:03   0:00 b6
nibble:  18348  8.2  0.1  1516   456 pts/1    R    03:03   0:00 b6
nibble:  18349  1.9  0.1  1516   456 pts/1    R    03:03   0:00 b6
nibble:  18350  1.7  0.1  1516   456 pts/1    R    03:03   0:00 b6
nibble:  18351  7.7  0.1  1516   456 pts/1    R    03:03   0:00 b6
nibble:  18352  7.3  0.1  1516   456 pts/1    R    03:03   0:00 b6
nibble:  18353  7.2  0.1  1516   456 pts/1    R    03:03   0:00 b6

```



# PGP

---

Pretty Good Privacy?

Brute Force

O famoso PGP!

---

Com certeza a ferramenta mais popular  
de criptografia acessível as massas!

Quem nunca usou o pgp?



Brute Force

## Por onde começar?

---

Uma idéia natural é fazer usar o próprio PGP com IPC para decryptar...

```
$ pgp -c file.pgp
```

File is conventionally encrypted.

You need a pass phrase to decrypt this file.

Enter pass phrase:

Brute Force

## PGP: problema encontrado

---

PGP faz um tratamento de sinais e possui uma interface com usuário que dificulta uso de IPC...

Brute Force

## PGP: soluções proposta

---

1. Eliminar tratamento de sinais e usar pipelines... (+ complexa)
3. Alterar interface para captura de senhas via arquivos... (+ simples)
5. Alterar interface para captura de senhas via argumentos... (+ eficiente)

# slave.c

Código do módulo mestre que controla os módulos escravo.

(...)

```
35     while((ch = getc(file)) != EOF) {
36         if (ch == '\n') {
37             go_fuck(pass);
38             memset(pass, 0, 20);
39         } else {
40             c[0] = ch;
41             c[1] = '\0';
42             strcat(pass, c);
43         } }

```

(...)

```
59     void go_fuck(char *pass) {
60         char cmd[64] = "/tmp/bf-pgp/fuck-pgp /tmp/bf-pgp/target.pgp ";
61         strcat(cmd, pass);
62         system(cmd);
63     }

```

(...)

# master.c

Código do módulo mestre que controla outras máquinas.

```
1 #include <stdio.h>
2 #include "pvm3.h"
3
4 main()
5 {
6     int info, bufid, arrived, i;
7     int end1, end2, arr1, arr2;
8     int cc1, cc2;
9     int tid1, tid2;
10    char buf1[100], buf2[100];
11    char master_name[100];
12
13    gethostname(master_name, 40);
14    printf("PGP brute force with PVM\n\n");
15    printf("[t%x] [%s] [master] Running brute force...\n", pvm_mytid(), master_name);
16
17    memset(buf1, 0, 100);
18    memset(buf2, 0, 100);
19
20    cc1 = pvm_spawn("slave", (char**)0, 0, "", 1, &tid1);
21    cc2 = pvm_spawn("slave", (char**)0, 0, "", 1, &tid2);
22
23    if (cc1 == 1) {
24        cc1 = pvm_recv(-1, -1);
25        info = pvm_bufinfo(cc1, (int*)0, (int*)0, &tid1);
26        if (info < 0)
27            printf("[t%x] [master] Something is wrong.\n", tid1);
28        pvm_upkstr(buf1);
29        printf("[t%x] %s\n", tid1, buf1);
30    } else printf("Error: cannot start slave.\n");
31
32    (...)
33 }
```

# master.c

Código do módulo mestre que controla outras máquinas.

```
(...)  
46 while(1) {  
47  
48     arr1 = pvm_probe(tid1, 2);  
49     if (arr1 > 0) {  
50         cc1 = pvm_recv(-1, -1);  
51         info = pvm_buinfo(cc1, (int*)0, (int*)0, &tid1);  
52         if (info < 0)  
53             printf("[t%x] [master] Something is wrong.\n", tid1);  
54         pvm_upkstr(buf1);  
55         printf("[t%x] %s\n", tid1, buf1);  
56         end1 = 1;  
57     }  
58  
59     arr2 = pvm_probe(tid2, 2);  
60     if (arr2 > 0) {  
61         cc2 = pvm_recv(-1, -1);  
62         info = pvm_buinfo(cc2, (int*)0, (int*)0, &tid2);  
63         if (info < 0)  
64             printf("[t%x] [master] Something is wrong.\n", tid2);  
65         pvm_upkstr(buf2);  
66         printf("[t%x] %s\n", tid2, buf2);  
67         end2 = 1;  
68     }  
69  
70     if ((end1 == 1) && (end2 == 1)) {  
71         break;  
72     } }  
(...)  
77 pvm_exit();  
78 exit(0);  
79 }
```

h2hc\_fuckpgp.avi

1' 02''

```
ASHLEY
nibble@ashley:/tmp/bf-pgp$ ls
README      compile.sh*  master*     slave*     target.$00  target.pgp  w?
check.sh*   fuck.pgp*   master.c    slave.c    target.$01  wl          wordlist
nibble@ashley:/tmp/bf-pgp$ cat target
Can you see?

nibble@ashley:/tmp/bf-pgp$
```

```
ALICIA
Syntax: help | command |
Commands are:
  add          Add hosts to virtual machine
  alias       Define/list command aliases
  conf       List virtual machine configuration
  delete     Delete hosts from virtual machine
  echo       Echo arguments
  export     Add environment variables to spawn ex
port list
  getopt     Display PVM options for the console t
ask
  halt      Stop pvmds
  help     Print helpful information about a com
mand
  id       Print console task id
  jobs    Display list of running jobs
  kill    Terminate tasks
  mstat  Show status of hosts
  names  List message mailbox names
  ps     List tasks
  pstat  Show status of tasks
  put    Add entry to message mailbox
  quit   Exit console
  reset  Kill all tasks, delete leftover mboxe
s
  setenv Display or set environment variables
  setopt Set PVM options - for the console tas
k *only*!
  sig    Send signal to task
  spawn  Spawn task
  trace  Set/display trace event mask
  unalias  Undefine command alias
  unexport  Remove environment variables from spa
wn export list
  version Show libpvm version
pvm> |
```

```
ALICIA
Slap by Slap,
running PGP brute force with PVM.

ATTENTION:
You must keep all file in /tmp/bf-pgp/

1. install and run pvm
2. extract bf-pgp tar.gz on /tmp
3. Run compile.sh
4. Run check.sh
5. Run bench.sh

Have fun,
Nibble <n1b0w[at]yahoo.com.br>
nibble@alicia:/tmp/bf-pgp$ bench.sh
PGP brute force with PVM
(benchmark test)

First test with 1.000 attempts...

real    0m2.601s
user    0m0.280s
sys     0m0.630s

Second test with 10.000 attempts...

real    0m25.146s
user    0m4.060s
sys     0m5.480s

Last test... Done!
nibble@alicia:/tmp/bf-pgp$
```



Força Bruta

## Primeiro Benchmark (1 hora)

---

# 2x

Maquinas de proc. Atholon XP 1.4 Ghz

256 Mb de ram

- **1.000.000 em cada máquina.**
- **total 2.000.000 de tentativas.**

Força Bruta

## Segundo Benchmark (1 horas)

---

# 3x

Maquinas com 4 proc. Intel Xeon 2.8 Ghz  
4 Gb de ram

- **3.000.000 em cada máquina.**
- **total 9.000.000 de tentativas.**

## Força Bruta Detalhe...

---

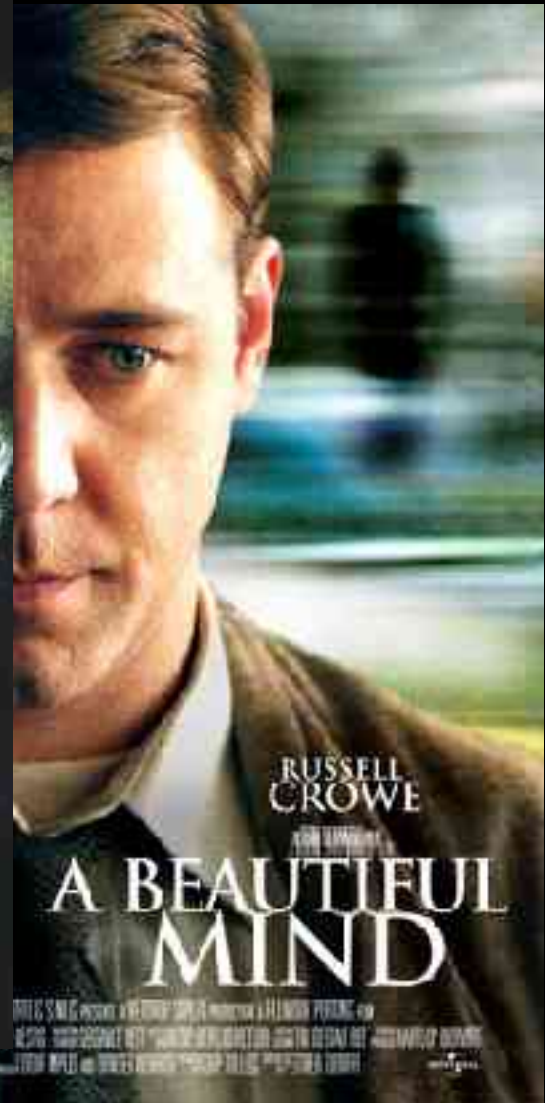
O programa foi desenvolvido sem muitas otimizações, faz referencias excessivas ao disco rígido. Considere que o mesmo ainda pode ser **BASTANTE** otimizado.

# Implicações Éticas

---

Pretty Good Privacy?

# Na mídia...



# Legal ou Illegal?

---

- ❑ O uso de criptografia é considerada ilegal para civis em diversos países (Israel, na França até 99).
- ❑ Felizmente, esquemas do tipo “key escrow” nunca obtiveram sucesso, alguém suspeita porque? ;)

Proteger-se do que?  
Proteger-se de quem?



# CYPHERPUNK CRIMINAL



<http://www.activism.net/cypherpunk/manifesto.htm>



# Referências na Internet

---

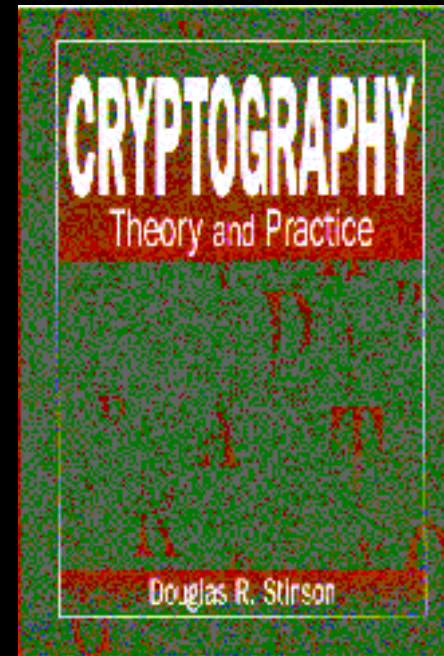
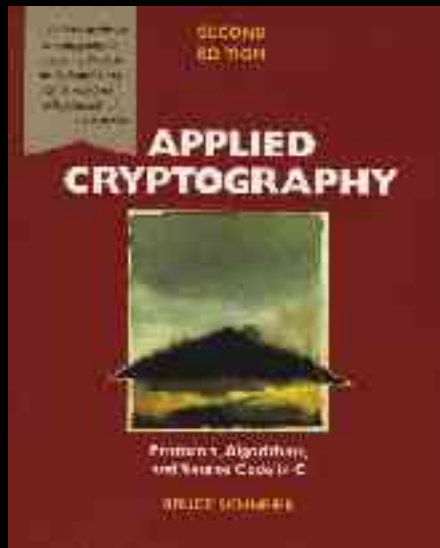
- A Cryptographic Compendium  
<http://home.ecn.ab.ca/~jsavard/crypto/intro.htm>
- RSA Laboratories FAQ  
<http://www.rsasecurity.com/rsalabs/node.asp?id=2152>
- LASEC, The Security and Cryptography Laboratory  
<http://lasecwww.epfl.ch/>

# Livros de Referência

---

## Applied Cryptography

[http://www.simonsingh.net/The\\_Code\\_Book.htm](http://www.simonsingh.net/The_Code_Book.htm)



## Cryptography, Theory and Practice

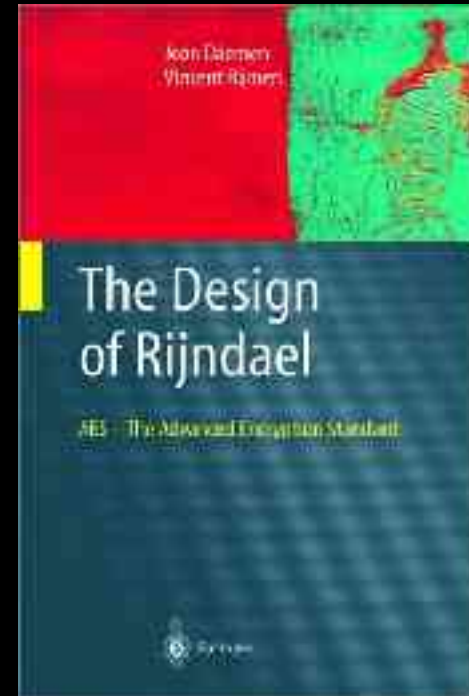
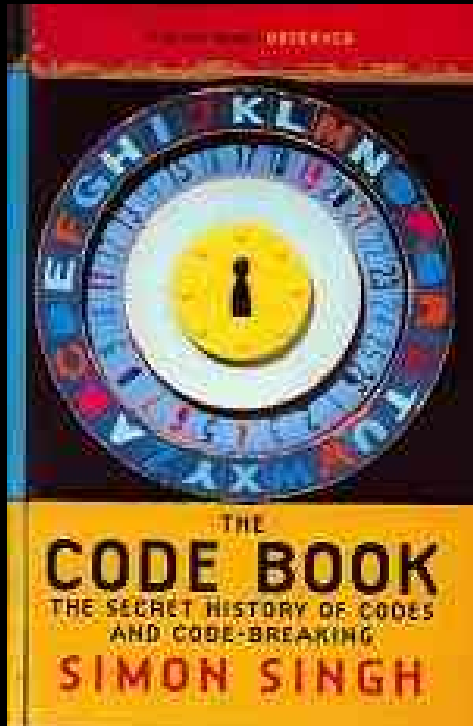
<http://www.cacr.math.uwaterloo.ca/~dstinson/CTAP.html>

# Livros de Referência

---

## The Code Book

<http://www.schneier.com/book-applied.html>



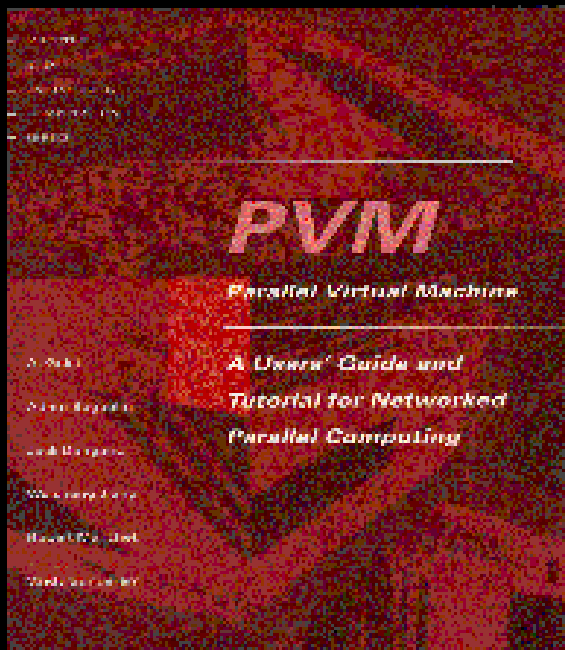
## The Design of Rijndael

<http://www.esat.kuleuven.ac.be/~rijmen/rijndael/>

# Livros de Referência

## PVM - Parallel Virtual Machine

<http://www.netlib.org/pvm3/book/pvm-book.html>



## MPI - Message Passing Interface

<http://www.netlib.org/utk/papers/mpi-book/mpi-book.html>

# Agradecimentos

---



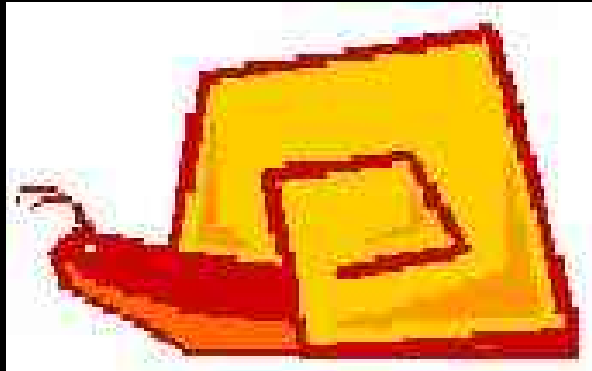
**Hackers 2 Hackers Conference**

<http://www.h2hc.com.br/>

# Agradecimentos

---

( pelos benchmarks )



**RFDS Labs**

<http://www.rfdslabs.com.br>

**“It is insufficient to protect ourselves  
with laws; we need to protect  
ourselves with mathematics.”**

>> Bruce Schneier in  
Applied Cryptography