UNIVERSIDADE DE ÉVORA

Sistemas Operativos II

Segurança e SD

Segurança

- introdução
- técnicas de segurança
- algoritmos de criptografia
- assinaturas digitais
- considerações sobre segurança

Introdução

- **Segurança**: requisito que se pretende num sistema com vista a garantir objetivos como:
  - autenticação
    - garantia sobre a identidade de um interveniente ou a origem de uma mensagem
  - integridade
    - não houve alteração na informação
  - confidencialidade/sigilo
    - o acesso à informação é feito apenas por intervenientes autorizados
  - não-repúdio
    - Garantia de que o envolvimento numa operação não pode ser posteriormente negado
  - assinatura digital
    - Atestado de ligação de uma entidade a um documento, para provar a sua origem, ou que a entidade teve conhecimento do respetivo conteúdo

Introdução

Conceitos:

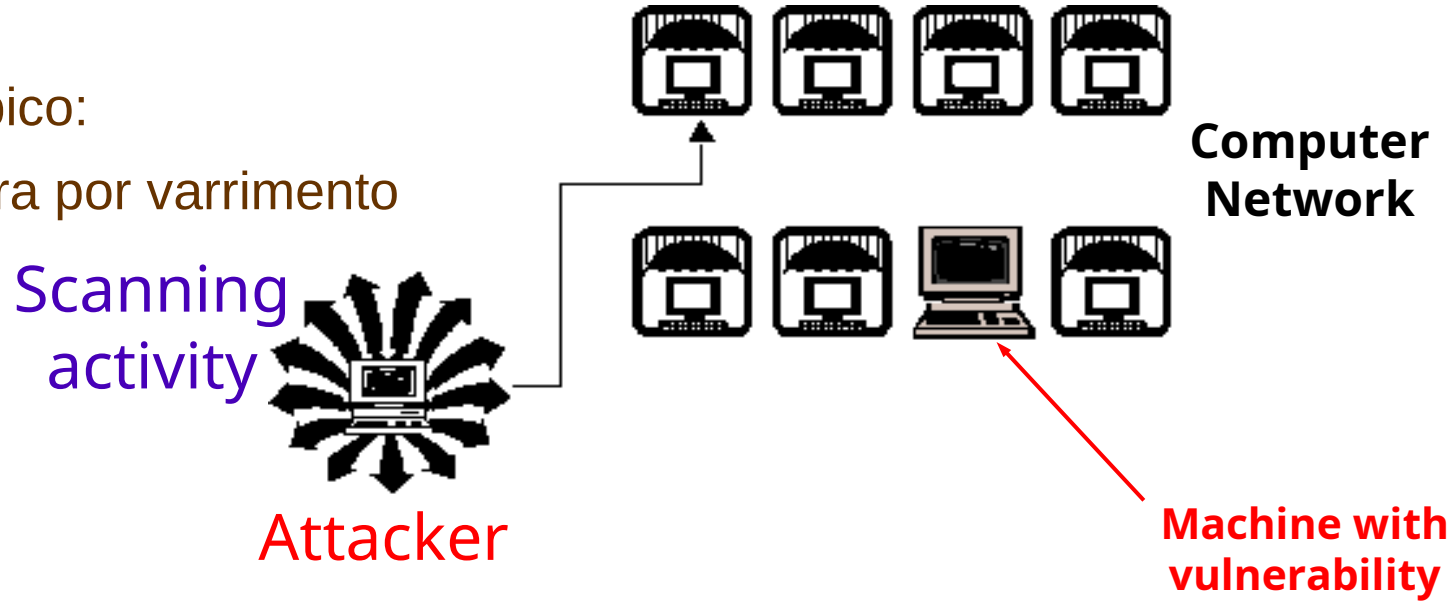
- política de segurança: **regulamento** para alcançar os objetivos de segurança
- mecanismo de segurança: **técnica** usada para implementar uma política de segurança (ex: fechadura, algoritmo de criptografia)

- *Principal*: entidade (utilizador ou processo) envolvida numa operação

- Criptografia é uma ciência que utiliza funções matemáticas para cifrar e decifrar dados
- Criptoanálise é a ciência da análise e quebra de segurança de sistemas baseados em criptografia
- Criptologia envolve Criptografia e Criptoanálise
- Chave é um parâmetro do algoritmo criptográfico

Intrusões (1)

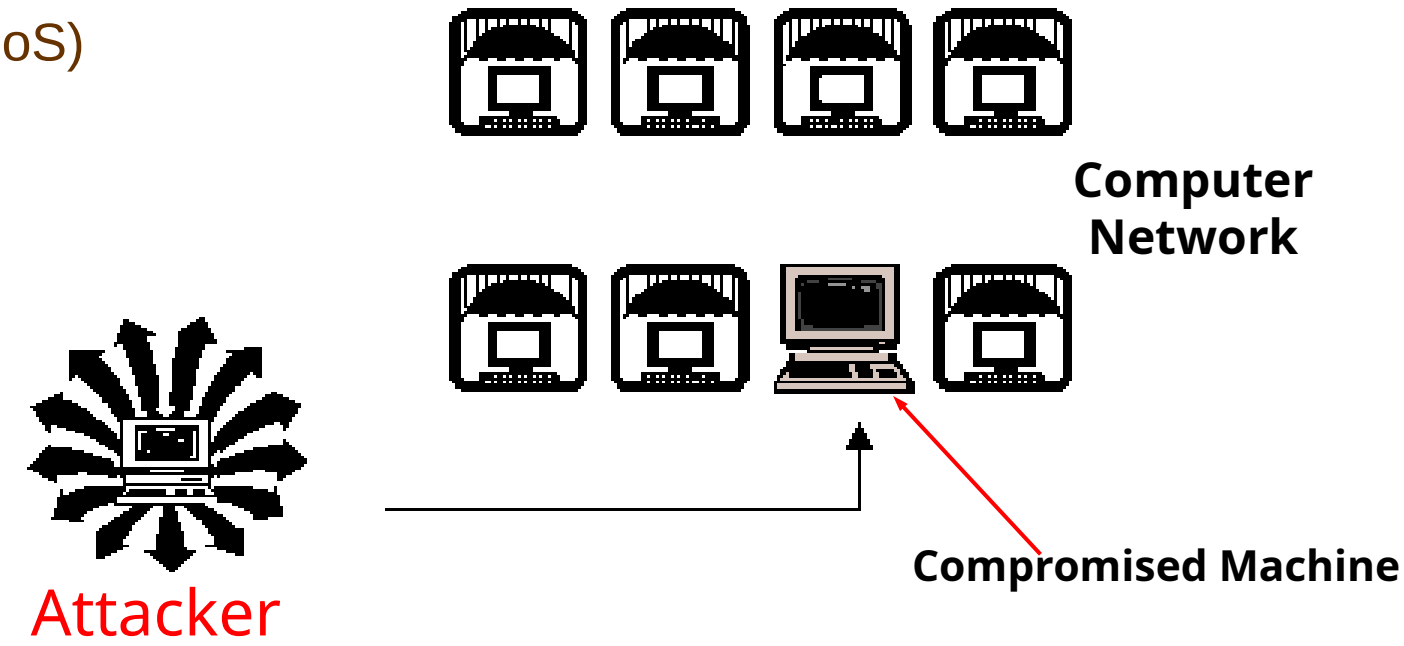
- Intrusões são ações para contornar os mecanismos de segurança em sistemas informáticos.
- Podem ser causados por:
  - Atacantes remotos que acedem pela rede
  - *Insiders* (utilizadores locais) – utilizadores do sistema que tentam ganhar privilégios ou fazer uso indevido das suas capacidades
- Cenário típico:
  - Procura por varrimento



Vipin Kumar, "Parallel and Distributed Computing for Cybersecurity," *IEEE Distributed Systems Online*, vol. 6, no. 10, 2005

Intrusões (2)

- Depois de detetada uma falha, o atacante tenta explorar
  - Acesso indevido a dados
  - Forjar identidades
  - Vandalismo (DoS)



Vipin Kumar, "Parallel and Distributed Computing for Cybersecurity," *IEEE Distributed Systems Online*, vol. 6, no. 10, 2005

Perspetiva Histórica

- Evolução das necessidades de segurança

	1965-75	*1975-89*	*1990-99*	*Current*
Platforms	Multi-user timesharing computers	Distributed systems based on local networks	The Internet, wide-area services	The Internet + mobile devices
Shared resources	Memory, files	Local services (e.g. NFS), local networks	Email, web sites, Internet commerce	Distributed objects, mobile code
Security requirements	User identification and authentication	Protection of services	Strong security for commercial transactions	Access control for individual objects, secure mobile code
Security management environment	Single authority, single authorization database (e.g. /etc/passwd)	Single authority, delegation, replicated authorization databases	Many authorities, no network-wide authorities	Per-activity authorities, groups with shared responsibilities

Ameaças de Segurança

- *leakage*
  - acesso à informação por um *principal* não autorizado
- *tampering*
  - alteração indevida da informação
- vandalismo
  - interferência ao funcionamento normal do sistema (sem que o causador/perpetrador tenha benefícios)

Ataques a um Sistema

ATAQUES onde se materializam uma ou mais ameaças

- *eavesdropping*
  - obter cópias de mensagens sem autorização
- *masquerading*
  - envio ou recepção de mensagens utilizando uma identidade de outro *principal* sem o seu consentimento
- *message tampering*
  - interceção e alteração de mensagens, que em seguida são enviadas para o destinatário original (Ex: *man-in-the-middle*)
- *replaying*
  - guardar uma mensagem interceptada para enviar mais tarde (pode funcionar mesmo com o uso de autenticação e cifra de mensagens)
- *denial of service*
  - congestionamento de um canal ou recurso para impedir o acesso por parte dos utilizadores comuns

Ameaças do Código Móvel

- código desconhecido executado localmente
- JAVA
  - cada ambiente tem um SecurityManager* que determina os recursos disponíveis para a aplicação
    - ex: nos browsers, as applets não podem aceder a ficheiros locais, etc.
  - Medidas de Proteção
    - o código remoto é separado do código local, para evitar trocas maliciosas
    - validação do código
- * Sem SecurityManager, a aplicação fica limitada ao código disponível na sua classpath.
- O SecurityManager assegura o cumprimento das regras:
  - `$JAVA_HOME/jre/lib/security/java.policy`
  - `$HOME/.java.policy`
  - ou um ficheiro de regras passado como argumento

Pressupostos e Princípios a seguir na implementação de sistemas seguros

Pressupostos a assumir:

- exposição de APIs
  - os SD têm interfaces de comunicação abertas que permitem ligações de novos clientes, logo, podem receber mensagens de um atacante
- redes inseguras
  - escuta, falsificação da fonte da mensagem
- algoritmos e código estão disponíveis para o atacante
- atacante tem recursos inesgotáveis

Princípios:

- limitar o período de utilização e âmbito de chaves secretas
- minimizar a base segura
  - a parte do sistema que implementa a segurança (hardware/software) deve ser fiável, logo, deve ter uma pequena dimensão.
  - ser céptico é um bom princípio

Técnicas de Segurança (1)

- Firewalls (filtros (origem, destino, porto, protocolo) aplicados ao tráfego da rede)

- Controlo de Acesso (de um processo a um recurso)
  - tabela de permissões verificada no servidor
  - *protection domain*
    - ***Capabilities*** (junto ao processo ou agente)
      - Dificuldade: remover/revogar privilégios já atribuídos
    - ***Access Control Lists*** (ACLs) (junto ao recurso)
      - Ex: permissões junto aos ficheiros em UNIX

```
drwxr-xr-x  gfc22  staff       264 Oct 30 16:57 Acrobat User Data
-rw-r--r--  gfc22  unknown       0 Nov  1 09:34 Eudora Folder
-rw-r--r--  gfc22  staff    163945 Oct 24 00:16 Preview of xx.pdf
drwxr-xr-x  gfc22  staff       264 Oct 31 13:09 iTunes
-rw-r--r--  gfc22  staff       325 Oct 22 22:59 list of broken apps.rtf
```

Técnicas de Segurança (2)

- Certificados, Credenciais
  - Elementos que atestam algo sobre quem o detém
    - Identidade?
    - Autorização?

- Criptografia, com o propósito de conseguir:
  - autenticação
  - integridade
  - confidencialidade
  - assinaturas digitais
  - não repúdio

– FILMES –

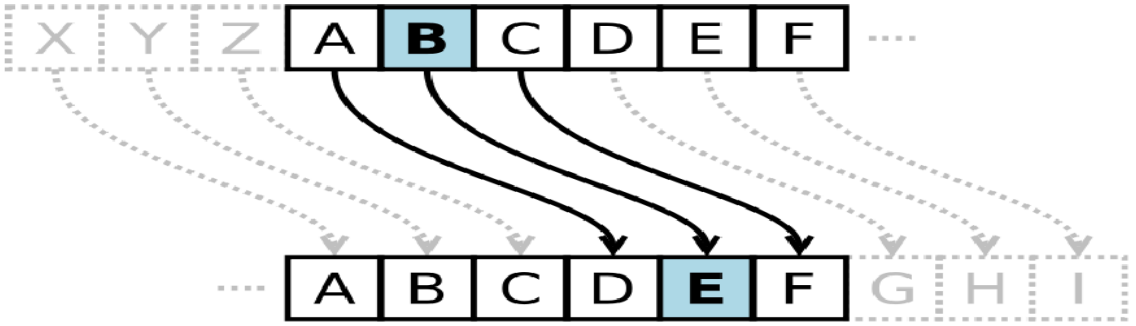
Criptografia: encriptação antiga

Scytale Transposition Cipher

- *O segredo é o diâmetro do cilindro*

Caesar Substitution Cipher

- *Rotação de N posições no alfabeto*



Criptografia: encriptação antiga

Vigenère Polyalphabetic Substitution

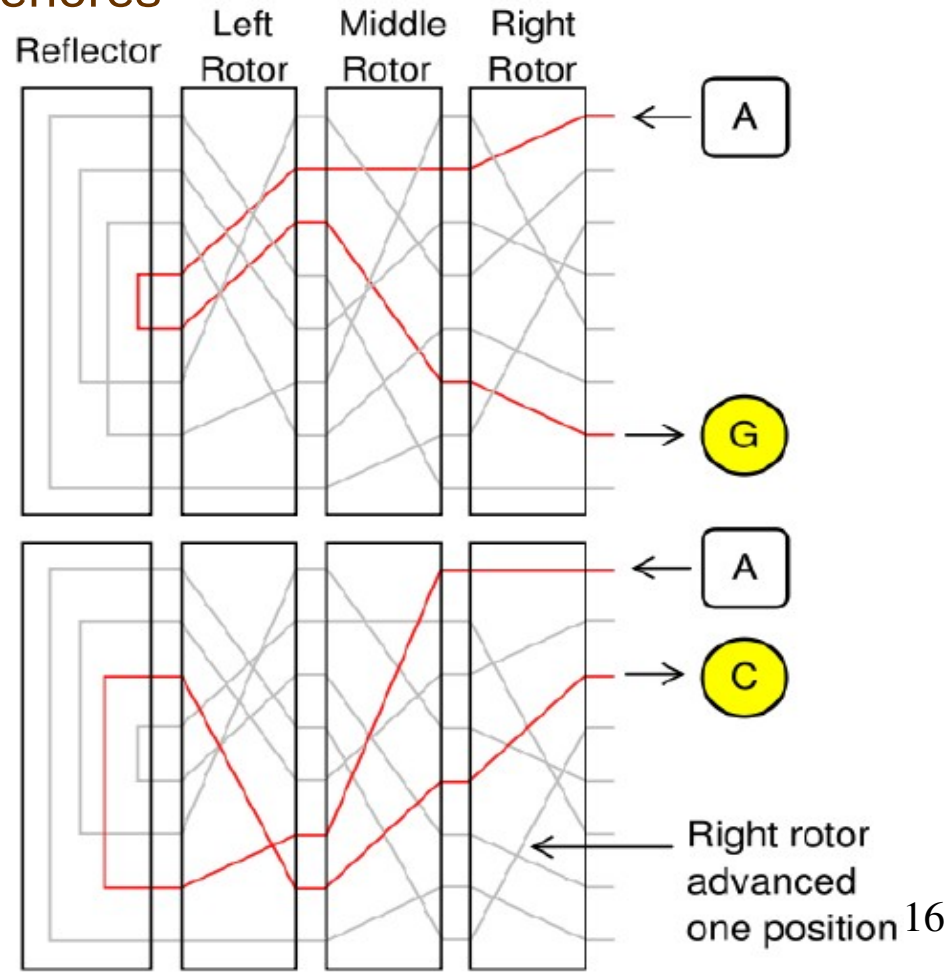
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T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

Key:
GOOGLE
Plaintext:
BUYYOUTUBE
Ciphertext:
HIMEZYZIPK

Criptografia: encriptação antiga

- Cifra baseada em mecanismos com rotores
  - Maior variabilidade que os anteriores

Enigma – *2ª Guerra Mundial*



Nomes tradicionalmente usados

- Nomes tradicionalmente usados na descrição de aspetos de segurança em cenários com vários participantes:

Alice	First participant
Bob	Second participant
Carol	Participant in three- and four-party protocols
Dave	Participant in four-party protocols
Eve	Eavesdropper
Mallory	Malicious attacker
Sara	A server

Principal (em Inglês) – interveniente num processo

Notações Criptográficas

- Convenção simbólica

K_A	Alice's **secret key**
K_B	Bob's secret key
K_{AB}	Secret key shared between Alice and Bob
K_{Apriv}	Alice's **private key** (known only to Alice)
K_{Apub}	Alice's **public key** (published by Alice for all to read)
$\{M\}_K$	Message M encrypted with key K
$[M]_K$	Message M signed with key K

Confidencialidade e Integridade: cenário 1

- Alice usa K_{AB} e $E(K_{AB},M)$, Bob usa $D(K_{AB},\{M\}_{KAB})$
  - a mensagem não é visível por terceiros
  - a encriptação garante a integridade se for adicionada informação (*checksum*) ao *ciphertext* que será verificada na desencriptação
  - problemas:
    - transmitir a chave secreta de Alice para Bob
    - Bob não consegue detetar uma mensagem capturada por Mallory e enviada mais tarde (*replaying*)

Autenticação

- o *principal* que decifra a mensagem com uma determinada chave pode assumir:
  - mensagem é autêntica se contem um *checksum* correto (se a operação de validação é bem sucedida)
  - a mensagem foi cifrada com a chave correspondente

Mais:

- Infere-se que o emissor da mensagem detinha a chave (secreta ou privada) correspondente para a encriptação
  - … deduzindo-se a sua **identidade**
    - assumindo que a chave não foi divulgada!!!

Algoritmos de Encriptação

- algoritmos usados para
  - transformar *plaintext* (mensagem ou dados no formato original) em *ciphertext* (dados codificados de modo ofuscado)
    - $E(K,M) = \{M\}_K$
  - o receptor do *ciphertext* aplica-lhe outra função do algoritmo para obter o *plaintext*
    - $D(K,E(K,M)) = M$

- **Tipo de Algoritmos de Encriptação:**
  - simétricos
  - assimétricos
  - outros: híbridos, block cipher, stream cipher

Algoritmos de Encriptação

- princípios relacionado com teoria da informação:
  - confusão
    - operações não destrutivas para ofuscar a relação entre um bloco plaintext e o respectivo ciphertext
  - difusão/dispersão
    - evitar a redundância, repetições (que poderiam dar pistas sobre a chave, numa analogia com o plaintext)

Algoritmos Simétricos

- chave secreta é partilhada (e escondida de todos os outros)
  - é argumento da função de encriptação E e da função de desencriptação D
- baseados em funções *one-way*
  - $F_k([M]) = E(K, M)$ fácil de calcular
  - função inversa $F^{-1}_k([M])$ tão difícil que é impraticável
- exemplo: DES

Algoritmos Simétricos: TEA

- Tiny Encryption Algorithm (TEA), Wheeler and Needham 1994
  - o plaintext é visto como sequência de blocos de 64 bits (2 inteiros 32 bits – *vector text[]*)
  - chave de 128 bits (4 inteiros de 32 bits)
  - usa
    - adição de inteiros (*linhas 4, 5, 6*)
    - bitwise logical shifts >> e << (*linhas 5, 6*), procura alcançar:
      - difusão
        - esconde repetição e redundância no *plaintext*
      - confusão
        - combina cada bloco do *plaintext* com a chave
        - ofusca a relação dos blocos em M com os de {M}K
        - evita a dedução da chave pela análise da frequência de caracteres no texto

Algoritmos Simétricos: TEA

- Função de encriptação

```
void encrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];                          1
    unsigned long delta = 0x9e3779b9, sum = 0; int n;                2
    for (n= 0; n < 32; n++) {                                        3
        sum += delta;                                                4
        y += ((z << 4) + k[0]) ^ (z+sum) ^ ((z >> 5) + k[1]);        5
        z += ((y << 4) + k[2]) ^ (y+sum) ^ ((y >> 5) + k[3]);        6
    }
    text[0] = y;  text[1] = z;                                       7
}
```

Exclusive OR

logical shift

Algoritmos Simétricos: TEA

- Função de desencriptação

```
void decrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];
    unsigned long delta = 0x9e3779b9, sum = delta << 5;  int n;
    for (n= 0; n < 32; n++) {
        z -= ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
        y -= ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
        sum -= delta;
    }
    text[0] = y; text[1] = z;
}
```

Algoritmos Simétricos: TEA

- aplicação...

```
void tea(char mode, FILE *infile, FILE *outfile, unsigned long k[]) {
/* mode is 'e' for encrypt, 'd' for decrypt, k[] is the key.*/
        char ch, Text[8]; int i;
        while(!feof(infile)) {
                i = fread(Text, 1, 8, infile);          /* read 8 bytes from infile into Text */
                if (i <= 0) break;
                while (i < 8) { Text[i++] = ' ';}       /* pad last block with spaces */
                switch (mode) {
                case 'e':
                        encrypt(k, (unsigned long*) Text); break;
                case 'd':
                        decrypt(k, (unsigned long*) Text); break;
                }
                fwrite(Text, 1, 8, outfile);            /* write 8 bytes from Text to outfile */
        }
}
```

Algoritmos Simétricos: TEA

- Performance
  - cerca de 3 vezes mais rápido que DES
- 128 bits na Chave
  - resistente contra ataques de força bruta

Algoritmos Simétricos: DES

- Data Encryption Standard (DES)
  - desenvolvido pela IBM, ANSI standard (1977)
  - encripta blocos de 64 bits de *plaintext em ciphertext* com igual tamanho
  - chave de 56 bits
  - encriptação
    - 16 “rondas” de rotação de bits
    - number of bits to be rotated determined by key plus 3 key-independent transpositions
  - algoritmo lento quando implementado em software, para os computadores da década de 70 e 80, mas rápido quando executado em *VLSI hardware* (*very large scale integration*)

Algoritmos Simétricos: DES

- DES foi quebrado/cracked pela 1ª vez em Junho de 1997
  - ataque de força bruta para descobrir a chave dado um para plaintext/cyphertext, e usá-la para decifrar uma mensagem encriptada
  - envolveu a participação de 14000 utilizadores de Internet, que correram uma aplicação em *background* nas suas máquinas
  - capacidade média estimada das máquinas: 200 MHz Pentium Processor
  - a chave foi descoberta em 12 dias, depois de se analisarem 25% dos 2^{56} valores possíveis (houve alguma sorte também!)
  - um segundo ataque em 1998, com hardware dedicado, levou 3 dias
  - os ataques recentes precisam de menos de 24 horas

- DES com chave de 56 bits pode considerar-se obsoleto

Algoritmos Simétricos: triple-DES

- triple-DES
  - aplica o DES por 3 vezes, com duas chaves
  - $E_{3DES}(K_1, K_2, M) = E_{DES}(K_1, D_{DES}(K_2, E_{DES}(K_1, M)))$
  - semelhante a um algoritmo simétrico comum com chave de 112 bits
  - mais resistente a ataques de força bruta que DES
  - desvantagem: má performance
    - são três operações DES

Algoritmos Simétricos: IDEA

- International Data Encryption Algorithm (IDEA)
  - desenvolvido em 1990, *Lai and Massey*
  - sucessor do DES
  - chave de 128 bits para encriptar blocos de 64 bits
  - baseado na álgebra de grupos
    - oito “rondas” de XOR, adição, multiplicação
  - como no DES, a mesma função serve para encriptar e desencriptar
    - uma vantagem quando se pretende implementar por hardware
  - mais seguro que o DES
  - 3 vezes mais rápido

Algoritmos Simétricos: AES

- Advanced Encryption Standard (AES)
  - resultou de “invitation for proposals” (US NIST 1997-2001)
  - Rijndael (Daemen and Rijmen*) algorithm
    - algoritmo baseado em iterações sobre blocos
    - comprimento variável para chaves e blocos
      - cada um pode ter, de forma independente: 128, 192 ou 256 bits de comprimento

- provavelmente o algoritmo simétrico mais utilizado

* - http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael-ammended.pdf

- … continua na próxima sessão