# Computer Security & Access Protection

# topics:

Overview and Terminology

Security requirements

Threats, adversaries and intruders

Attacks from outside the system

Attacks from inside the system

Security holes

Protection mechanisms

Trusted systems

# Trust
# Security
# Protection

# ?

# translation of terms:

| | |
|---|---|
| Authenticity: | Authentizität |
| Availability: | Verfügbarkeit |
| Confidentiality: | Vertraulichkeit |
| Denial of Service: | Dienstverweigerung |
| Integrity: | Datenintegrität, Schutz gegen unautorisierte Veränderung |
| Intruder, Adversary: | Eindringling, Angreifer, Gegner |
| Privacy: | Datenschutz |
| Protection: | Zugriffsschutz |
| Security: | (Informations-) Sicherheit (Betriebssicherheit= safety) |
| Security threat: | Bedrohung |
| Trust: | Vertrauenswürdigkeit |

# Definitions:

**Trust** is a property within a social organization with respect to handling information. Trust defines the requirements and the resulting policies defined by an application area concerning the proper usage of information in the temporal and functional domain. It reflects the flow of information in an organization and is specified in terms of rules between authorization of subjects and clearance of information.

**Security** is the property of an information processing system. Security defines the requirements useful for an owner and user of information to protect it against security threats. Basic requirements which have to be assured in spite of intentional and malicious attacks are the confidentiality, integrity, availability and authenticity of information.

**Protection** is the set of hardware and software mechanisms to enforce security in a system.

# Access Control

Trusted System:

Mandatory access control.

Rules defined by organization policy.

---

Secure System:

Discretionary, user defined access control.

Rules defined by individual user.

Goal: Flexibility, Expressiveness, Least Privilege.

---

Protection System:

Mechanisms in the hardware and the operating system to enforce access specifications.

# Security vs. Privacy

Security protects data against misuse by individuals.

Privacy protects individuals against the misuse of data.

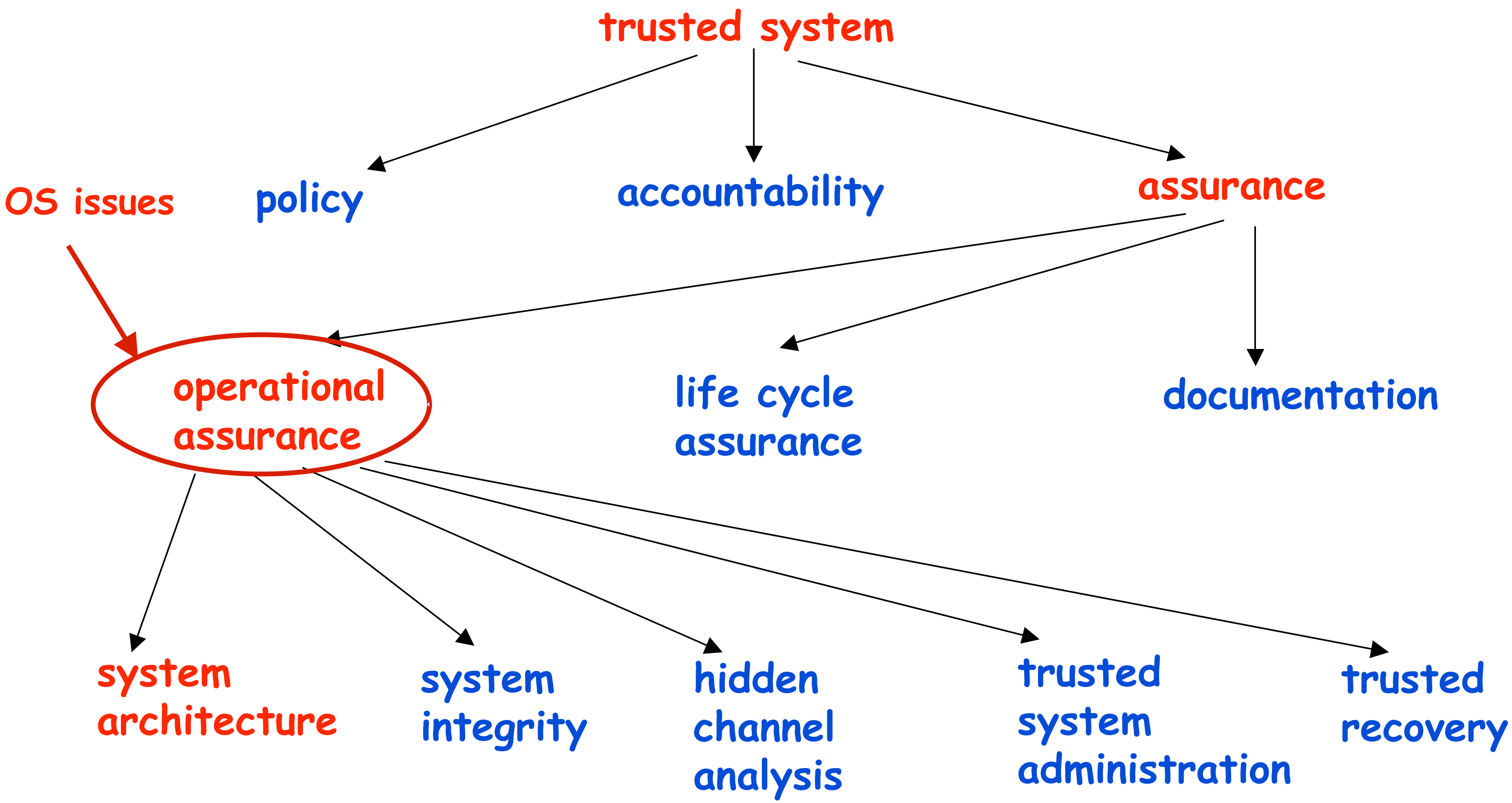
Security is a necessary but not a sufficient condition for trust and privacy !

# requirements for security

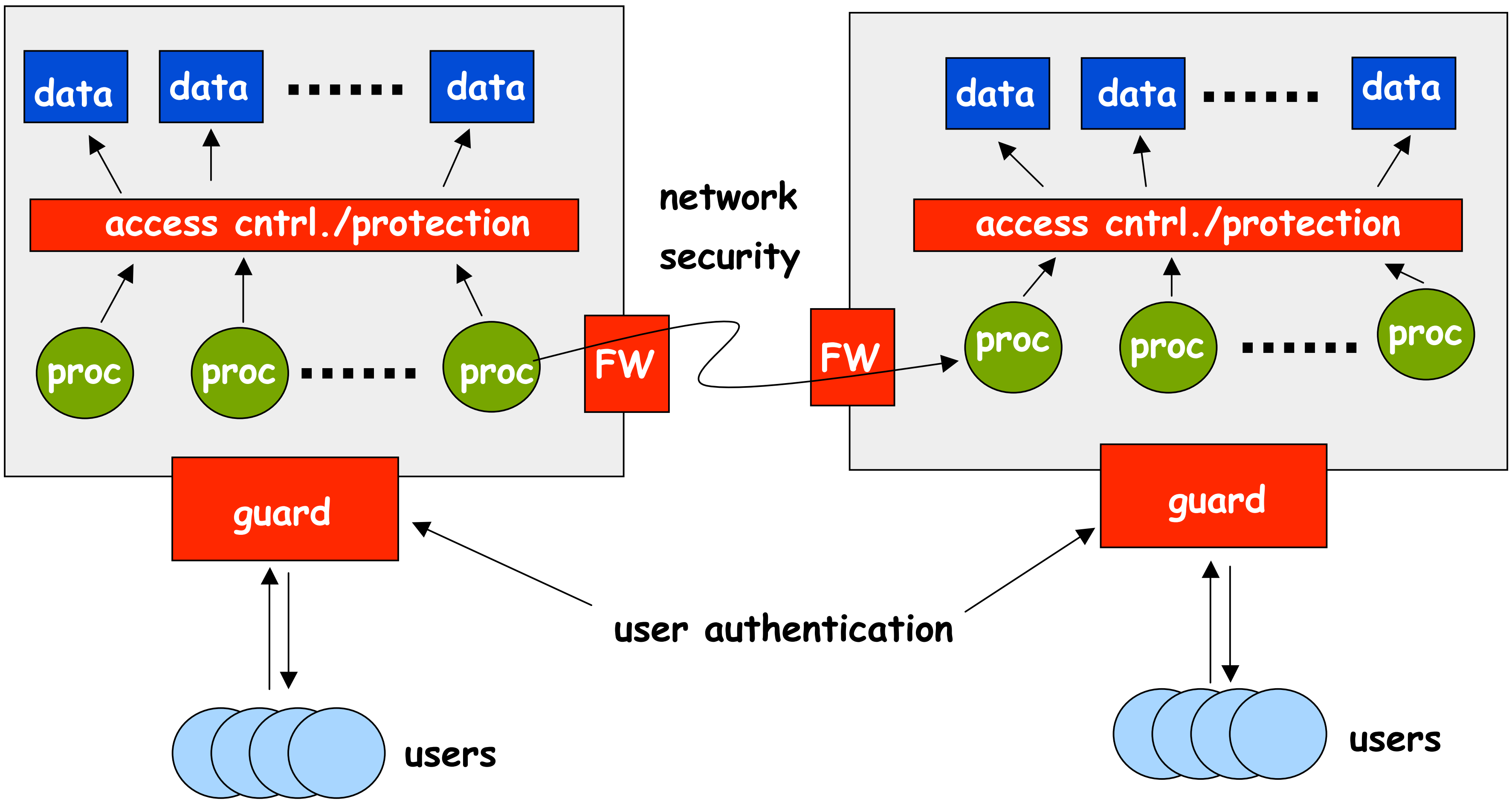
| | |
|---|---|
| Confidentiality: | data should not be read by unauthorized parties. |
| Integrity: | data should not be changed by unauthorized parties. |
| Availability: | data should be accessible when they are needed. |
| Authenticity: | the identity of subjects may not be forged |

# structuring requirements

acc. DoD Orange Book

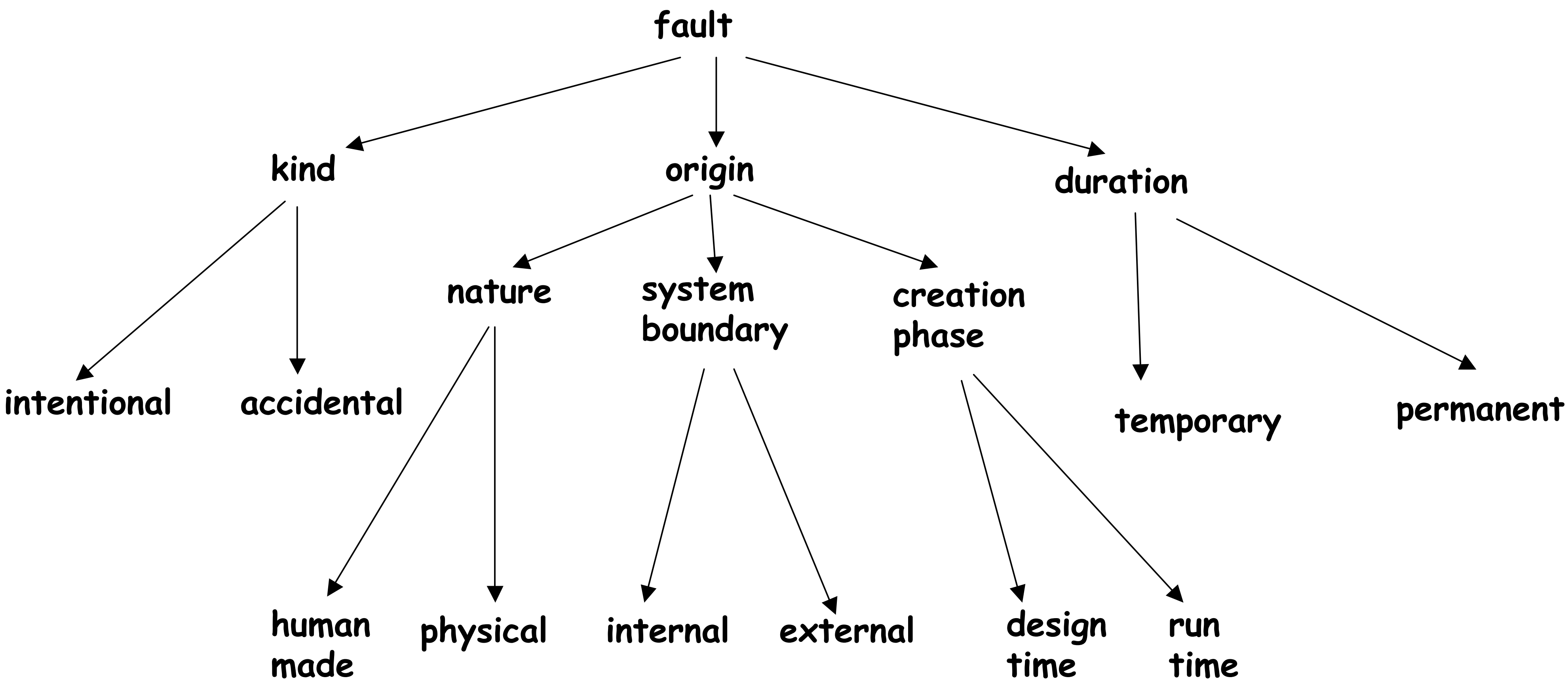


# system vulnerabilities



# classification of threats

a threat emerges from a fault in some system component or a fault by some user of the system




acc. J.C. Laprie: Dependability: Basic Concepts and associated teminology, 1990

# classification of threats

## example 1: threats caused by intentional (malicious), human-made faults

| system boundary | | creation phase | | duration | | threat |
|---|---|---|---|---|---|---|
| internal | external | desing time | run time | perm. | temp. | |
| | x | | x | x | | Intrusion |
| | x | | x | | x | Intrusion |
| x | | | x | x | | Virus |
| x | | x | | x | | Trojan Horse |
| x | | x | | x | | malicious logic |

# classification of threats

## example 2: threats caused by accidental faults

| | system boundary | | creation phase | | duration | | threat |
|---|---|---|---|---|---|---|---|
| | internal | external | desing time | run time | perm. | temp. | |
| physical | x | | | x | x | x | denial of service |
| | x | | | x | x | x | loss of integrity |
| | x | | | x | x | x | loss of confidentiality |
| human made | x | | x | | x | | loss of integrity |
| | x | | x | | x | | loss of confidentiality |

by software or
hardware design faults

# classification of adversaries

- occasional non-expert intruders
- expert insiders, unauthorized experienced hackers hacking the system
- expert insiders which have authorized access to the system
- espionage (military and company systems)
- higher forces: Fire, flood, earthquakes
- faults and bugs in the computer and the network
- just humans: e.g. disk with highly confidential data on the garbage etc.

# Authentication

# &

# Access Protection

**Authentication:** ensures that a user is the one she pretends to be. Authentication is based on a secret or on a non-forgable Identifier.On a standard OS, authentication is enforced by the “Login” procedure.

On the OS-Level, processes act on behalf of a user. A process usually obtains the privileges which are granted for the user.

**Access Protection:** ensures that an authenticated users (or the processes acting on behalf of the user) only have access to exactly the items they are allowed to use. This is enforced by memory and file protection mechanisms.

# attacks from outside of the system

## The login procedure

```
LBL>telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD:root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL
```

Stoll 89

**Tabelle 15.2** Beobachtete Passwortlänge

| Länge | Anzahl | Anteil an Gesamtheit |
|---|---|---|
| 1 | 55 | 0,004 |
| 2 | 87 | 0,006 |
| 3 | 212 | 0,02 |
| 4 | 449 | 0,03 |
| 5 | 1.260 | 0,09 |
| 6 | 3.035 | 0,22 |
| 7 | 2.917 | 0,21 |
| 8 | 5.772 | 0,42 |
| Gesamt | 13.787 | 1,00 |

From: William Stallings: Betriebssysteme, Prinzipien und Umsetzung, 4. Auflage, Pearson Studium, 2003

**Tabelle 15.3** Passwörter, die aus einer Probe von 13.797 Konten geknackt wurden [KLEI90]

| Passwortart | Suchgröße | Anzahl der Treffer | Erratene Passwörter in Prozent |
|---|---|---|---|
| Benutzer-/Kontoname | 130 | 368 | 2,7% |
| Zeichenfolge | 866 | 22 | 0,2% |
| Zahlen | 427 | 9 | 0,1% |
| Chinesisch | 392 | 56 | 0,4% |
| Ortsnamen | 628 | 82 | 0,6% |
| Gebräuchliche Namen | 2,239 | 548 | 4,0% |
| Frauennamen | 4,280 | 161 | 1,2% |
| Männernamen | 2,866 | 140 | 1,0% |
| Ungewöhnliche Namen | 4,955 | 130 | 0,9% |
| Mythen und Legenden | 1,246 | 66 | 0,5% |
| Shakespearesch | 473 | 11 | 0,1% |
| Sportbegriffe | 238 | 32 | 0,2% |
| Science-Fiction | 691 | 59 | 0,4% |
| Filme und Schauspieler | 99 | 12 | 0,1% |
| Comics | 92 | 9 | 0,1% |
| Berühmte Menschen | 290 | 55 | 0,4% |
| Redewendungen und Muster | 933 | 253 | 1,8% |
| Nachnamen | 33 | 9 | 0,1% |
| Biologie | 58 | 1 | 0,0% |
| Wörterbuch des Systems | 19.683 | 1,027 | 7,4% |
| Rechnernamen | 9.018 | 132 | 1,0% |
| Mnemonik | 14 | 2 | 0,0% |
| King James-Bibel | 7.525 | 83 | 0,6% |
| Verschiedene Wörter | 3.212 | 54 | 0,4% |
| Jiddische Wörter | 56 | 0 | 0,0% |
| Asteroide | 2.407 | 19 | 0,1% |
| GESAMT | 62.727 | 3,340 | 24,2% |

**From:** William Stallings:Betriebssysteme, Prinzipien und Umsetzung, 4. Auflage, Pearson Studium, 2003

# Def. One-Way-Function

Definition: One-Way Function

Informally, a function $f$ is a one-way function if

1. The description of $f$ is publicly known and does not require any secret information for its operation.
2. Given $x$, it is easy to compute $f(x)$.
3. Given $y$, in the range of $f$, it is hard to find an $x$ such that $f(x) = y$

   More precisely, any efficient algorithm solving a P-problem succeeds in inverting $f$ with negligible probability.

The existence of one-way functions is not proven. If true, it would imply $P \neq NP$. Therefore, it would answer the complexity theory NP-problem question of whether all apparently NP-problems are actually P-problems. Yet a number of conjectured one-way functions are routinely used in commerce and industry. For example, it is conjectured, but not proved, that the following are one-way functions:

1. Factoring problem for randomly chosen primes $p$, $q$.
2. Discrete logarithm problem.
3. Discrete root extraction problem. This is the function commonly known as RSA encryption.
4. Quadratic residue problem.

**Used e.g. in password encryption, Public Key Cryptography, Digital Signatures, ...**

Eric W. Weisstein. "One-Way Function." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/One-WayFunction.html

**Scientific American August 1977, Martin Gardener, Column "Mathematische Spiele".**

**He claimed that it would take "millions of years" to break the code.**

**N=114 318 625 757 888 867 669 235 779 976 146 612 010 218 296 721 242 362 562 561 842 935 706 935 245 733 897 839 597 123 563 958 705 058 989 075 147 599 290 026 879 543 541**

**26.November 1994 the factors were found by a group of 600 volunteers.**

**Range: $10^{129}$**

**Simon Singh: "The Code Book", 1999**

# passwd security

/etc/passwd holds a list of <name, encoded passwd>

passwd guessing: prepare a list of common passwd, encoded passwd
read the /etc/passwd from some computer
compare encoded passwd
on match > store <name, passwd>

salt: create entries: <name, random number, encoded passwd>
to obtain a match, the cracker has to generate $b^n$ (b=base n=exponent) versions of each passwd.

better passwd: longer names, not in a dictionary, numbers, special characters

one-time passwd: only used once. (Lamports algrithm to generate the list)

# One-time password

Initialize with P0 = f(P1)

It is only possible to calculate an old, already used password.

P1 = f(f(f(f(s))))

P2 = f(f(f(s)))

P3 = f(f(s))

P4 = f(s)

↑ Pn-1 is easy to calculate

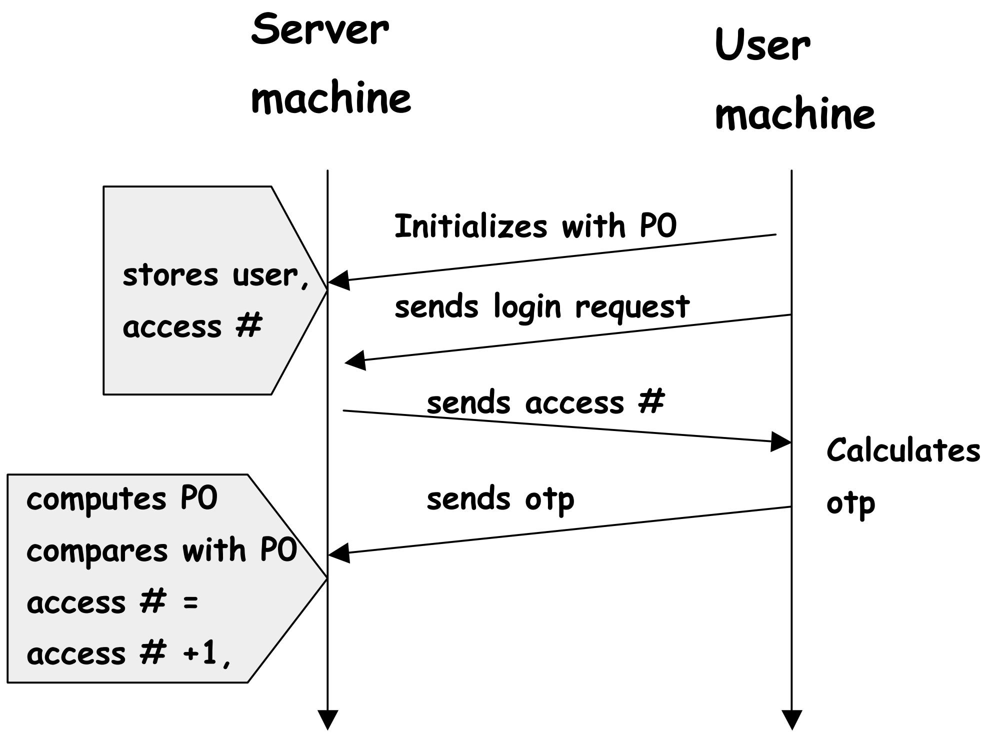
↓ Pn+1 is difficult to calculate

Only the user knows the secret s.

P0 is stored in the computer.

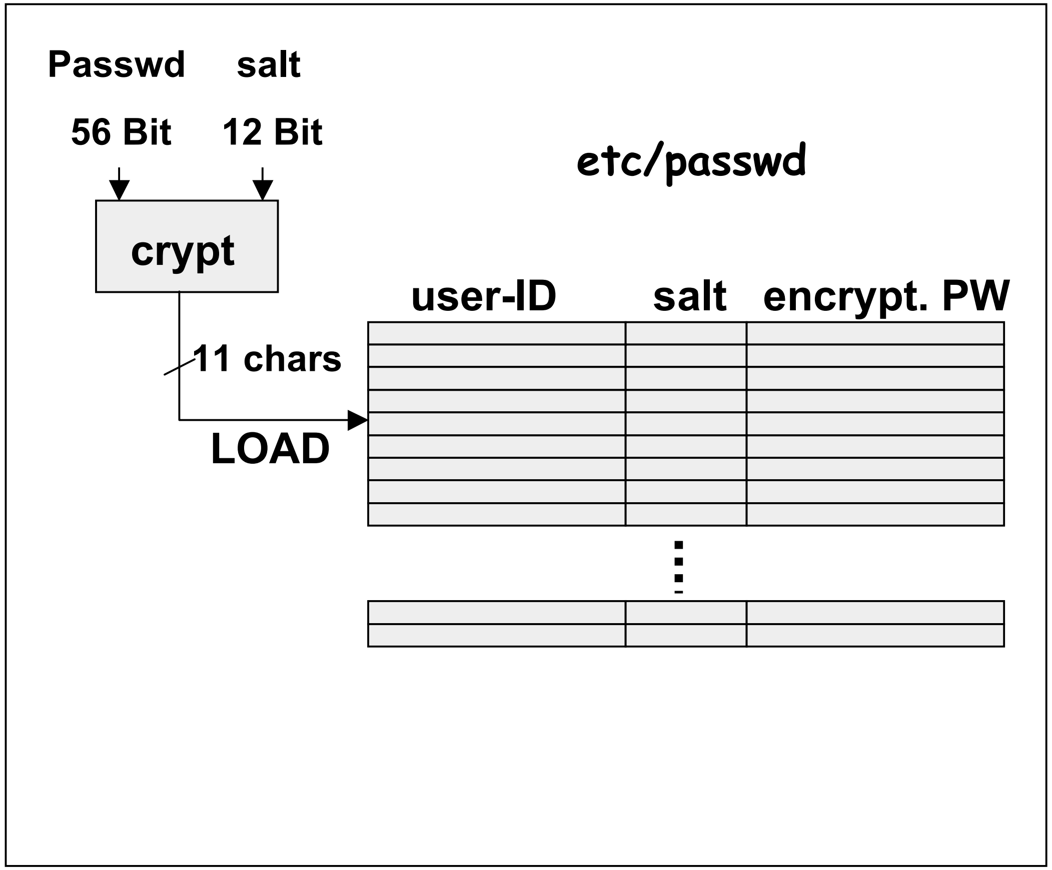
Pn is given by the user. The system applies the function f n times and compares the result with P0. If it matches, login is granted.

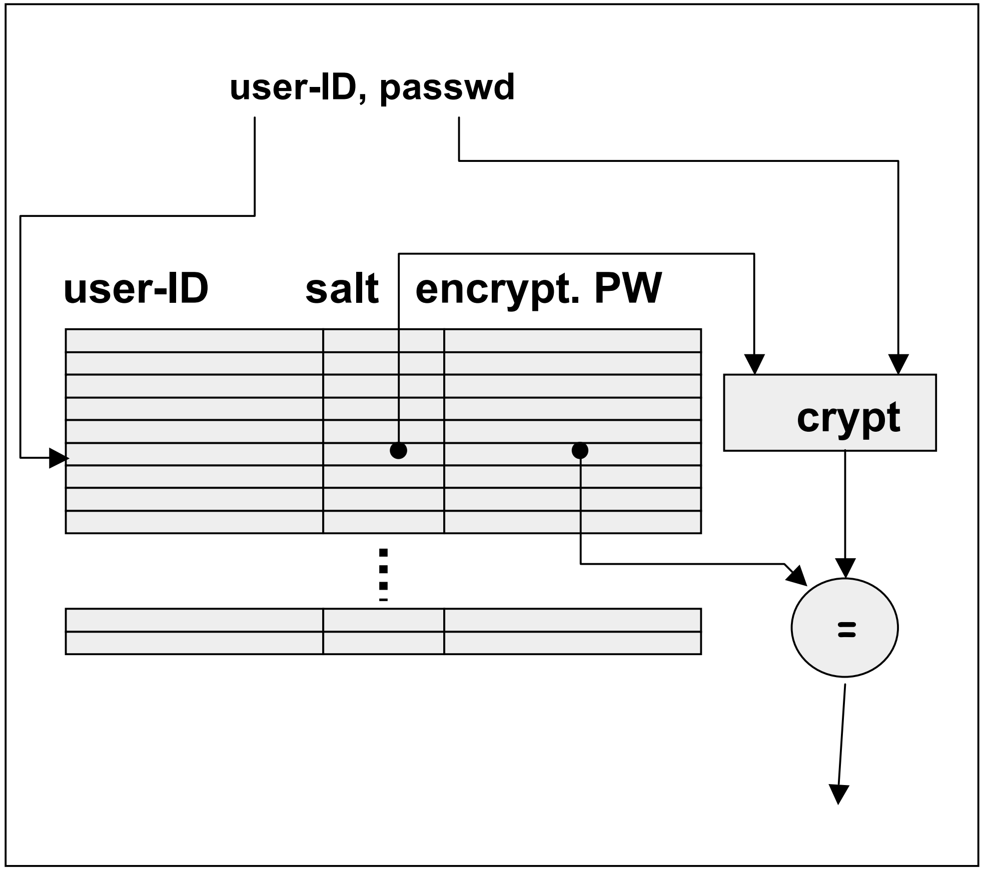
# One-time password




Basic
Protocol

# Password mechanisms in Unix



# more authentication

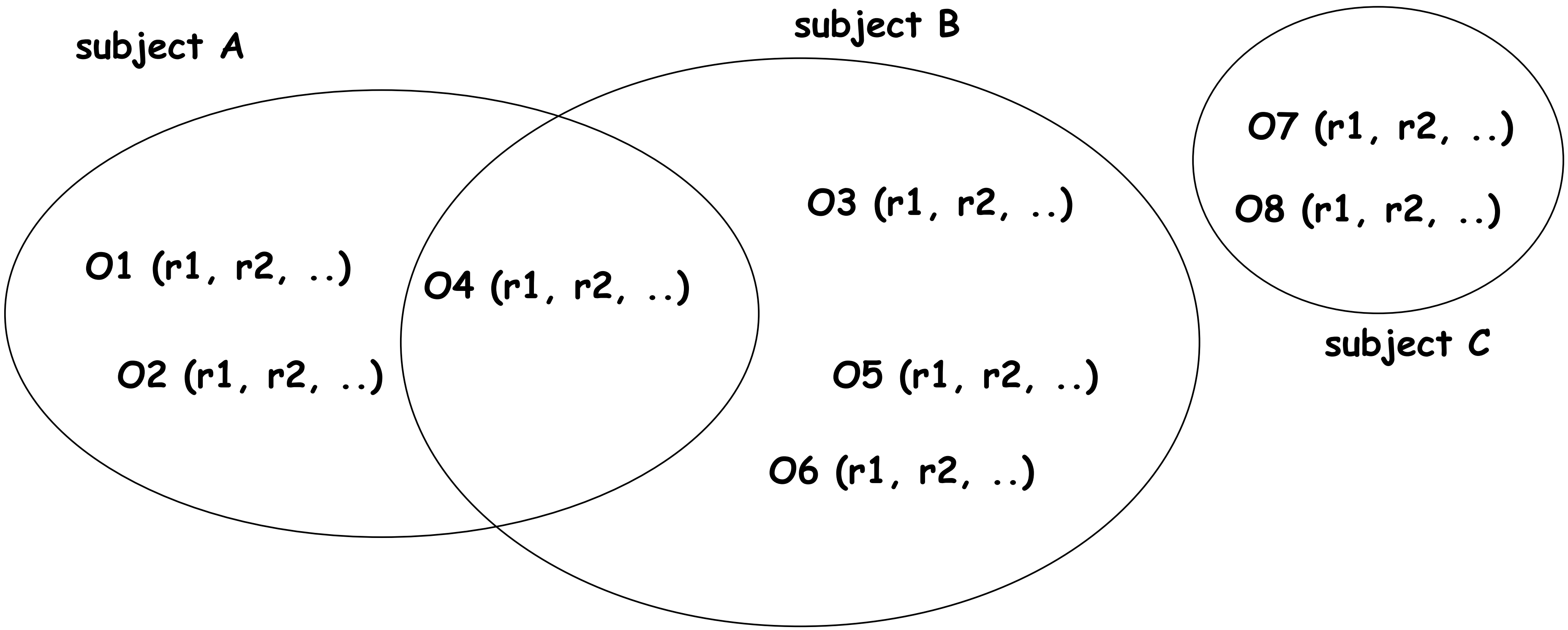
challenge-response

chip card + PIN

- magnetic (~ 140 Bytes, costs 0,1 -0,5 €)
- memory cards (~1 KB, ~1 €)
- smart cards ( 8bit CPU, 16 KB ROM, 4 KB EEPROM, 512 Bytes RAM, 9600 bps communication channel)

biometric authentication

# Protection mechanisms in the OS

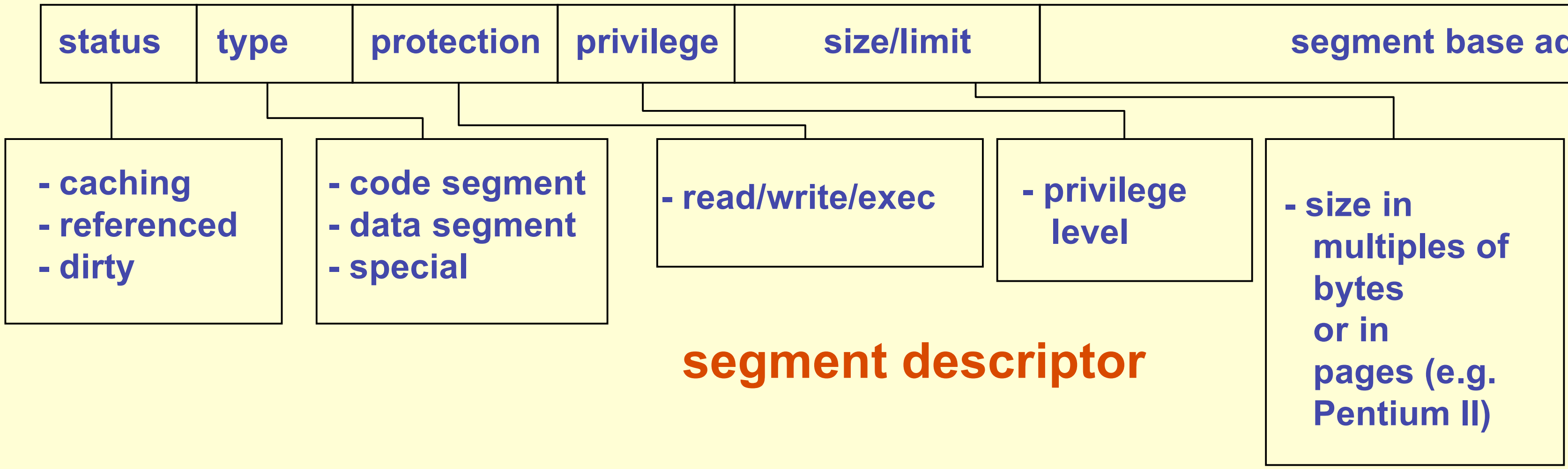



Protection Domains define the acess relations between

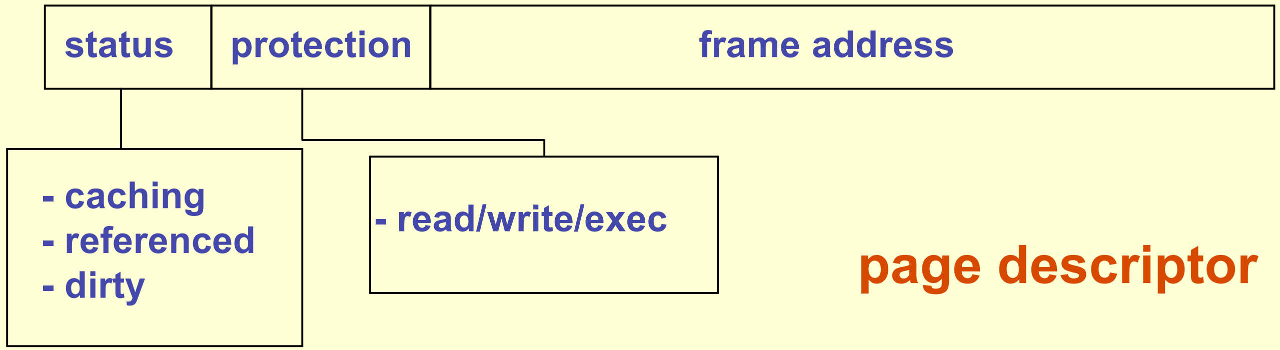
Active system components: Subjects, e.g. users, processes,.. and

Passive system components: Objects, e.g. files, devices, ...

# memory protection




segment descriptor




page descriptor

# access protection

**Lampson's model:**

paper "Protection" first appeared in *Proc. 5th Princeton Conf. on Information Sciences and Systems*, Princeton, 1971, p 437.

entities are distinguished as:

subjects, taking the active role in the system, and

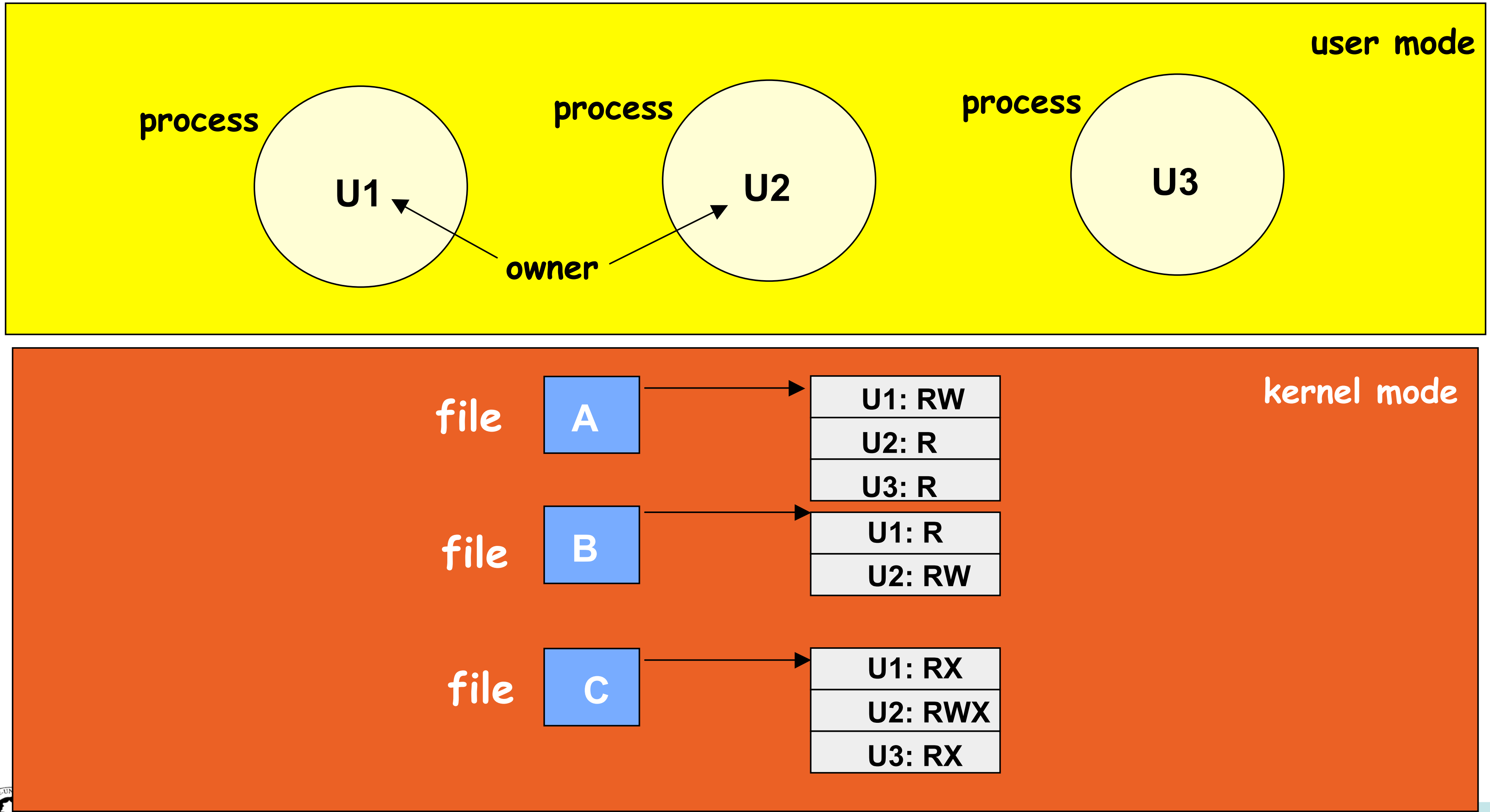
objects, that are passive entities

capability list for $s_3$

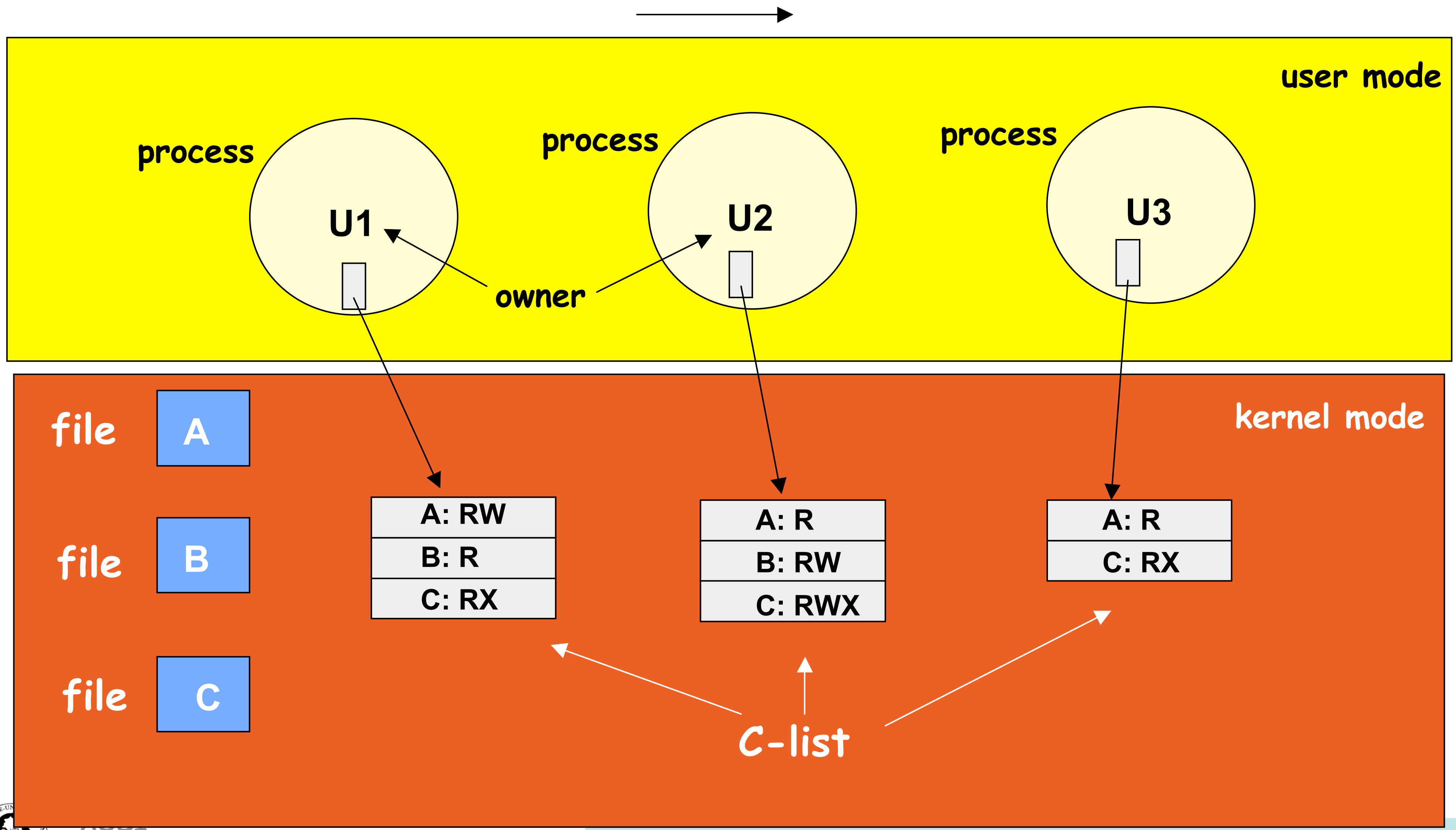
| subjects / objects | $s_1$ | $s_2$ | $s_3$ | | $s_k$ | | $s_n$ |
|---|---|---|---|---|---|---|---|
| $o_1$ | $R(o_1,s_1)$ | $R(o_1,s_2)$ | $R(o_1,s_3)$ | | $R(o_1,s_k)$ | | $R(o_1,s_n)$ |
| $o_2$ | $R(o_2,s_1)$ | $R(o_2,s_2)$ | $R(o_2,s_3)$ | | $R(o_2,s_k)$ | | $R(o_1,s_n)$ |
| | | | | | | | |
| $o_j$ | $R(o_j,s_1)$ | $R(o_j,s_2)$ | $R(o_j,s_3)$ | | $R(o_j,s_k)$ | | $R(o_1,s_n)$ |
| | | | | | | | |
| $o_m$ | $R(o_m,s_1)$ | $R(o_m,s_2)$ | $R(o_m,s_3)$ | | $R(o_m,s_k)$ | | $R(o_1,s_m)$ |

ACL for $o_2$

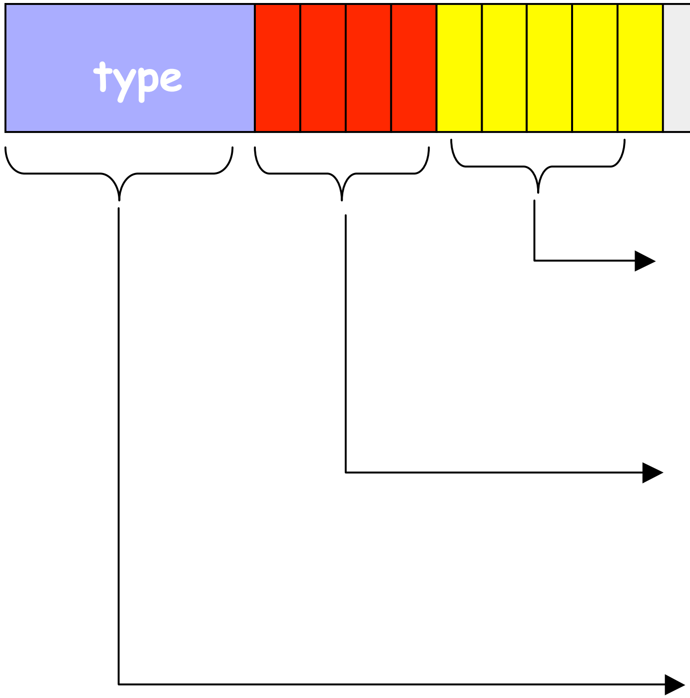
# Access Control List (ACL)



# Capability List (C-List)



# structure of a capability




rights defined for the object:

e.g.: read, write, excute,..., other type specific rights

rights defined for the capability itself:

e.g.: read, write, copy, transfer

type of the capability:

refers to the diffrent system levels on which the cap. is used, e.g. segment cap, file cap, user object..

# Discussion:

how to protect capabilities?

- Tagging
- Separation
- Encryption
- Sparse name space

More Problems:

- Controlling and confining capability transfer.
- Revocation of rights

(contradiction of terms according to Roger Needham (1992))

# Discussion: ACL vs. C-Lists

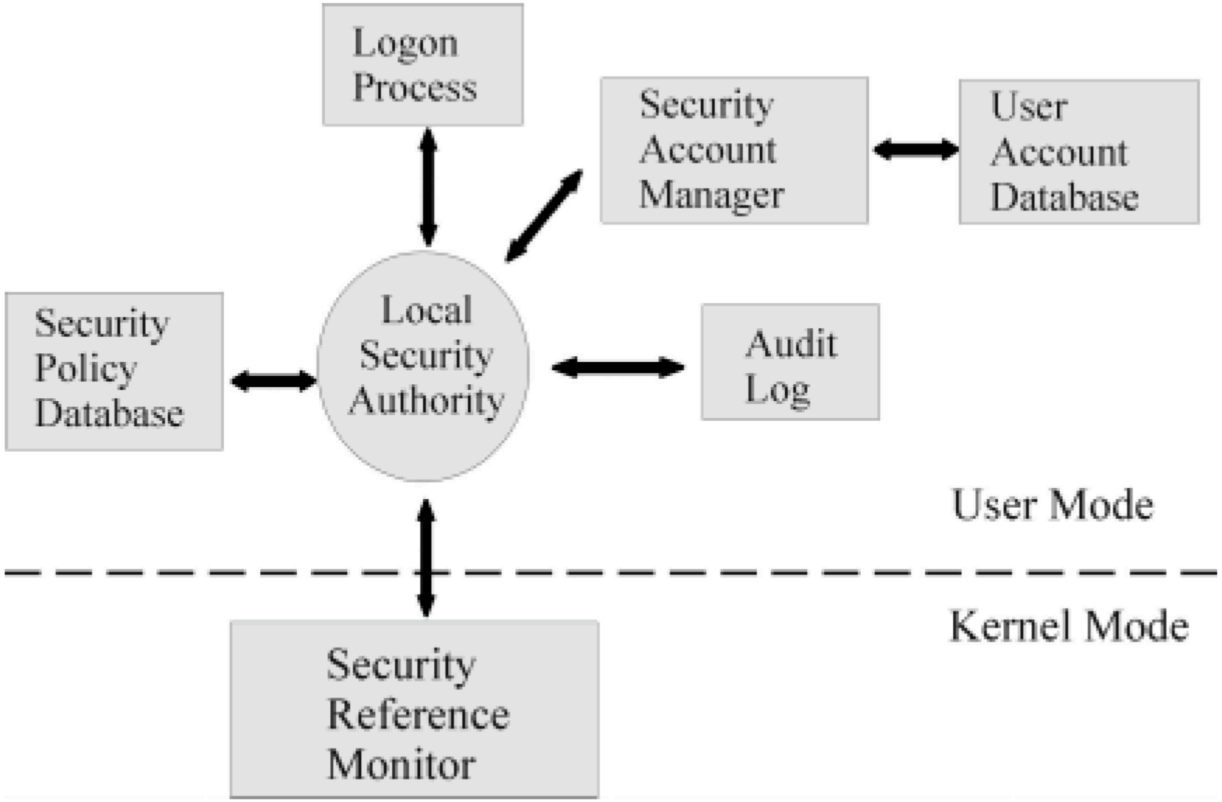
| | ACL | C-List |
|---|---|---|
| General mechanism | list based | ticket based |
| Authentication | every access | on capability creation |
| Addressability | unrestricted | confined to objects in the C-List |
| Referencing of objects | extra mechanism | combined mechanism |
| Transfer of rights | not possible | regulated by specifi rigths |
| Revocation of rights | easy | problem (possibly not desirable) |
| Granularity of objects | large objects | small objects |

# Security and access protection in W2K

- secure login and antispoofing
- discretionary access control
- privileged access control
- process address space protection
- prevention of data leaks by zeroing all new pages before loading
- security auditing

# overall NT security model

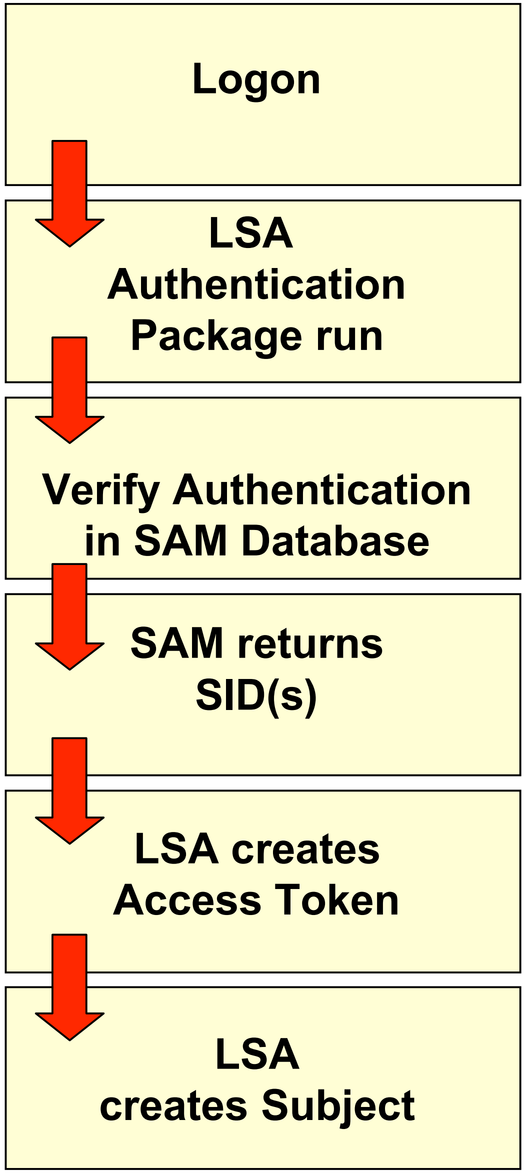
http://www.ciac.org/ciac/documents/CIAC-2317_Windows_NT_Managers_Guide.pdf



# NT logon process

Windows NT logon processes provide mandatory logon for user identification and cannot be disabled.

To protect against spoofing, the logon process begins with a Welcome message box that requests the user to press Ctrl, Alt and Del keys before activating the actual logon screen.



# the access token

| header | expir. time | groups | standard DACL | owner SID | group SID | restricted SIDs | privileges |
|---|---|---|---|---|---|---|---|

**Security ID (SID):** The SID is a variable length unique name (alphanumeric character string) that is used to identify an object, such as a user or a group of users in a network of NT/2000 systems.

**Expiration time:** defines validity interval for the access token (currently not used)

**Discretionary Access Control List (D ACL):** Default ACL when they are created by a process and no other ACL is specified.

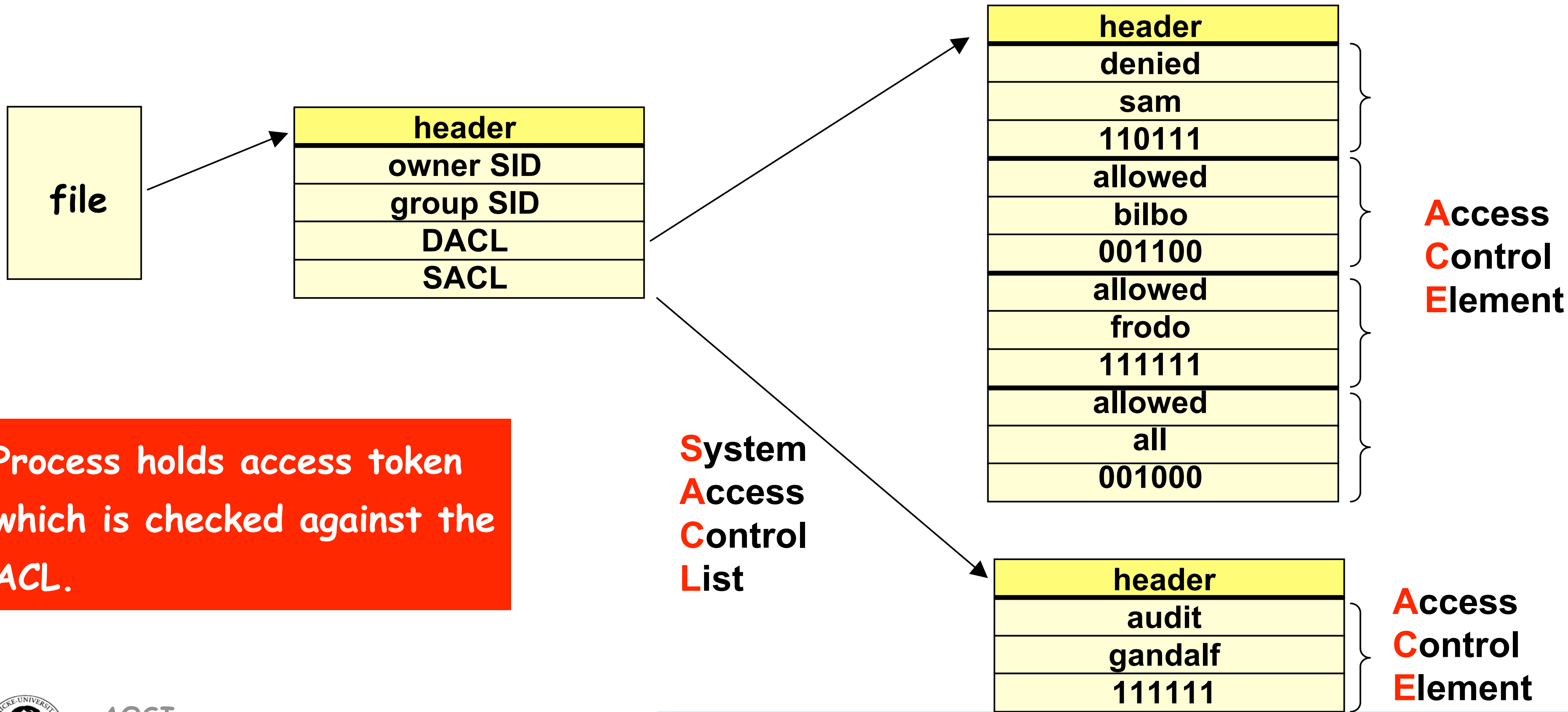
**Owner/group SID:** indicates the user/group who owns the process.

**Restricted SID:** enables the cooperation of trusted and non-trusted processes by constraining access for the latter.

**Privileges**: enable to define "admin rights" in a more fine-grained fashion and associate these with user processes.

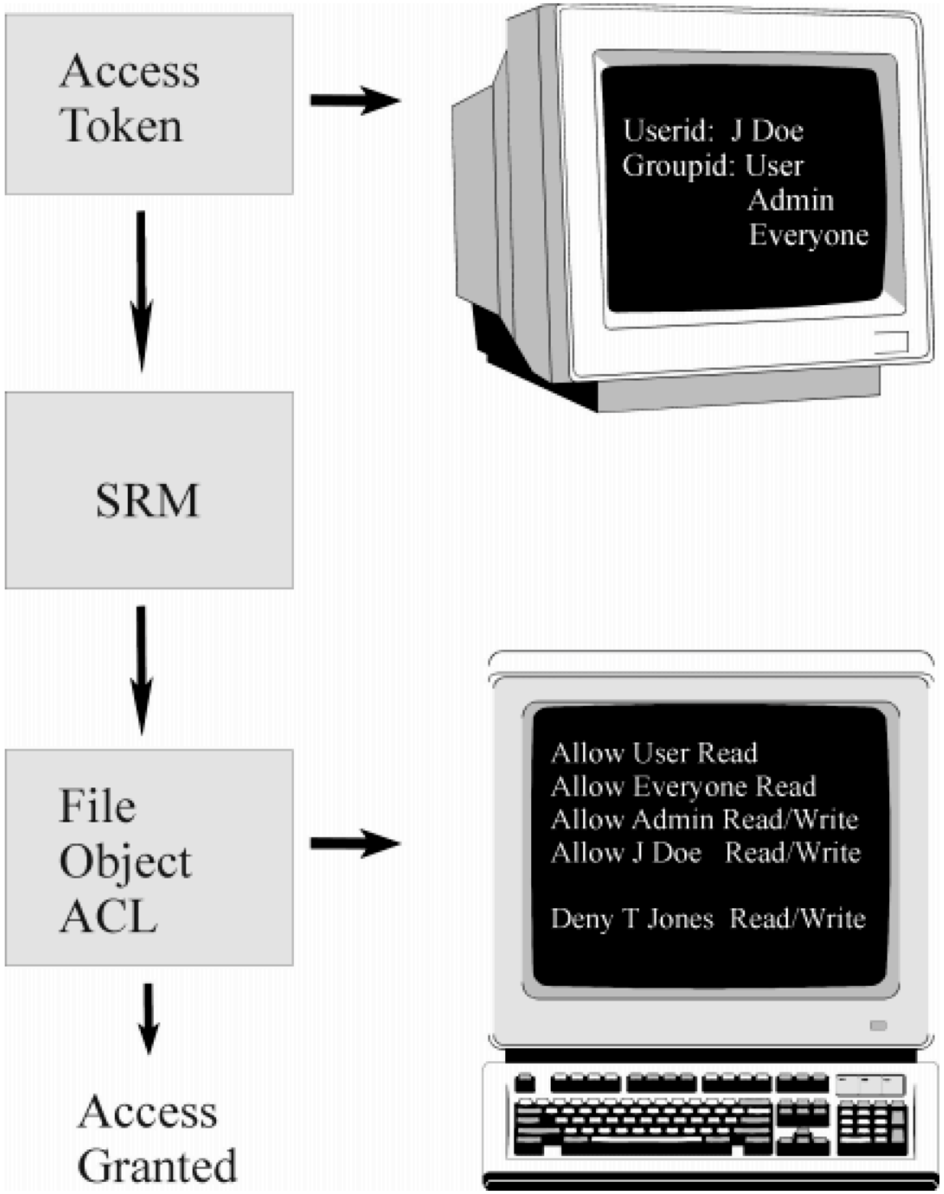
# the security descriptor

- is associated with every object
- defines who may access the object with which operation




Process holds access token which is checked against the ACL.

# SRM access validation



# what cryptography can do for security

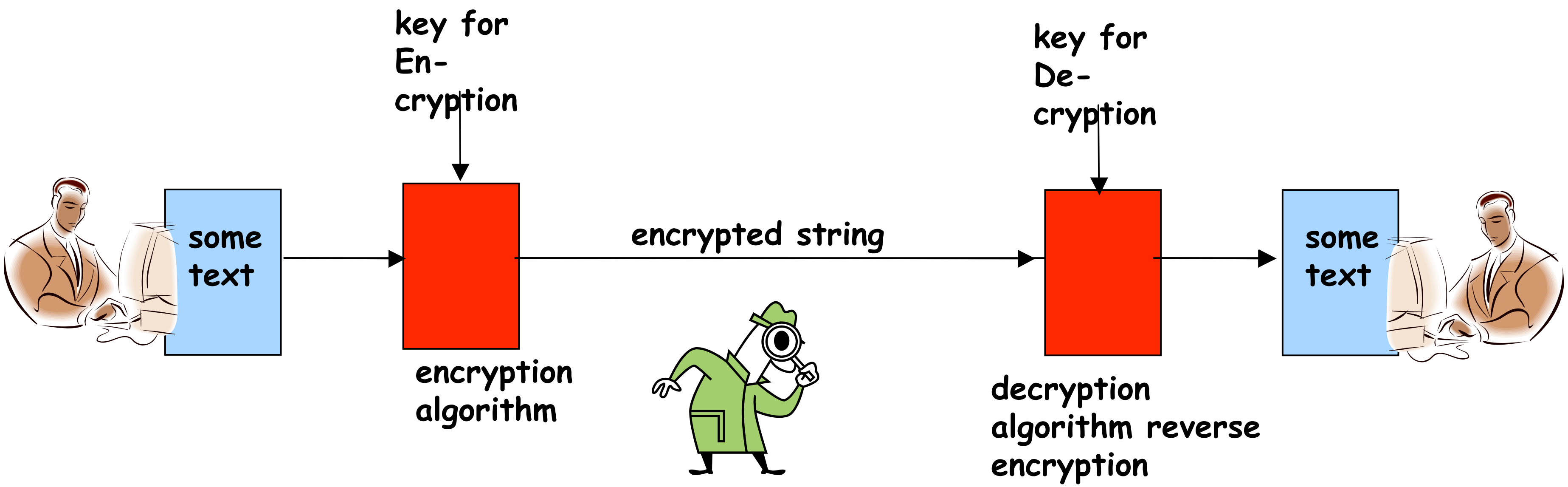
**Useful for transmission and storage of data !!**

| | |
|---|---|
| Confidentiality | encryption of data |
| Integrity | encryption, digital signatures |
| Authenticity | encryption of authentication information |

**Mechanisms:**

- one-way functions
- cryptographic hash funtions
- symmetric cryptosystems with a secret key (DES)
- asymmetric cryptosystems with a combination of public/secret key

# Securing the data transfer



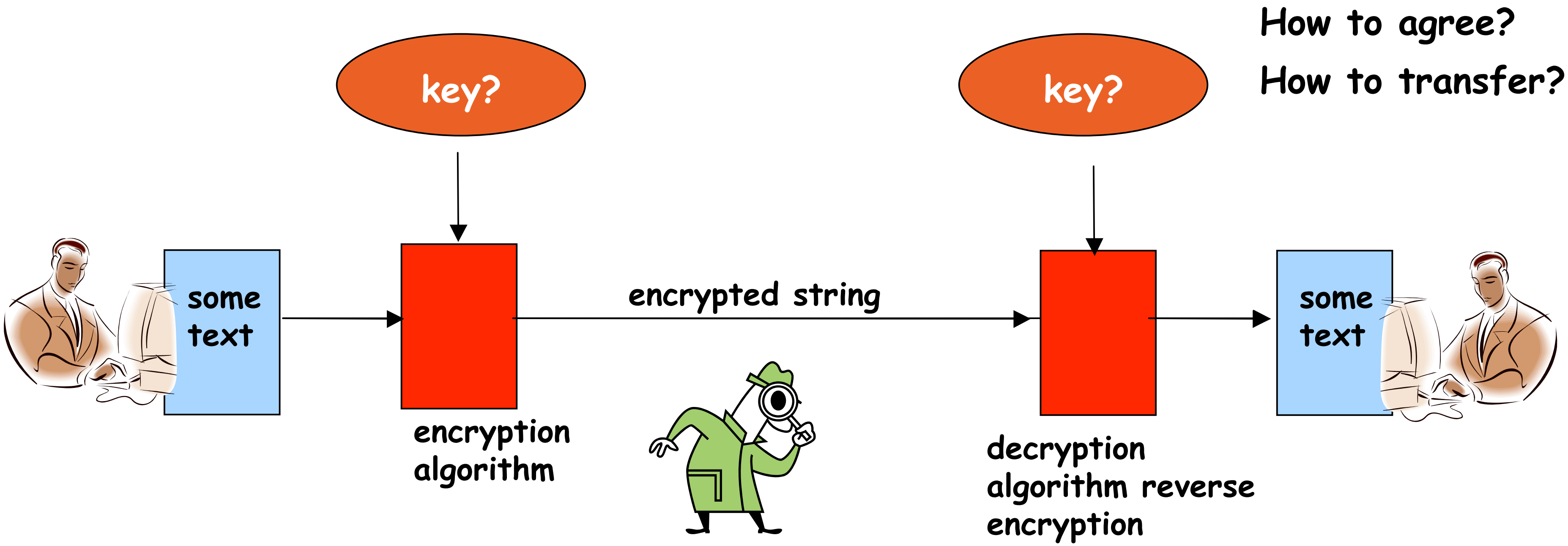

Eavesdropping,

Message faking,

Message corruption,

Masquerade.

# Securing the data transfer




Eavesdropping,

Message faking,

Message corruption,

Masquerade.

So far:

how to create a key not exchanging a secret?

Hellman, Diffie, Merkle

how to simplify key exchange and distribution?

Rivest, Shamir, Adleman

# Def. Cryptographic Hash-Function

A ***hash function H*** is a transformation that takes an input m and returns a fixed-size string, which is called the hash value *h* (that is, *h* = *H*(*m*)). Hash functions with just this property have a variety of general computational uses, but when employed in cryptography, the hash functions are usually chosen to have some additional properties.

The basic requirements for a **cryptographic hash function** are as follows.
The input can be of any length.
The output has a fixed length.
*H*(*x*) is relatively easy to compute for any given *x*.
*H*(*x*) is one-way.
*H*(*x*) is collision-free.

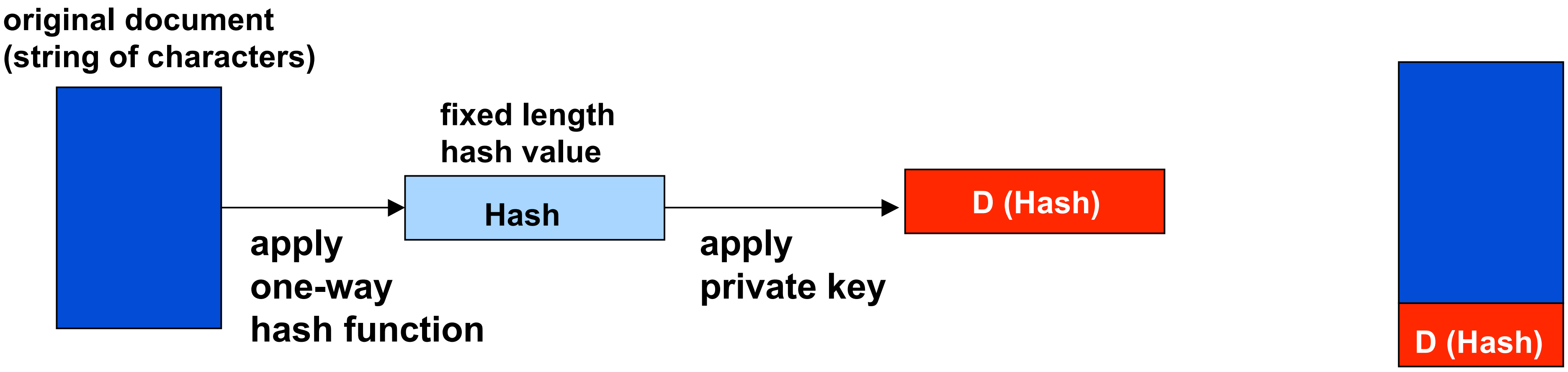
A hash function *H* is said to be ***one-way*** if it is hard to invert, where ``hard to invert'' means that given a hash value *h*, it is computationally infeasible to find some input *x* such that *H*(*x*) = *h*.

If, given a string *x*, it is computationally infeasible to find a string *y* not equal to *x* such that *H*(*x*) = *H*(*y*), then *H* is said to be a ***weakly collision-free*** hash function.

A ***strongly collision-free*** hash function *H* is one for which it is computationally infeasible to find any two strings *x* and *y* such that *H*(*x*) = *H*(*y*).

**(http://www.rsasecurity.com/rsalabs/node.asp?id=2176)**

# Example: Digital Signatures



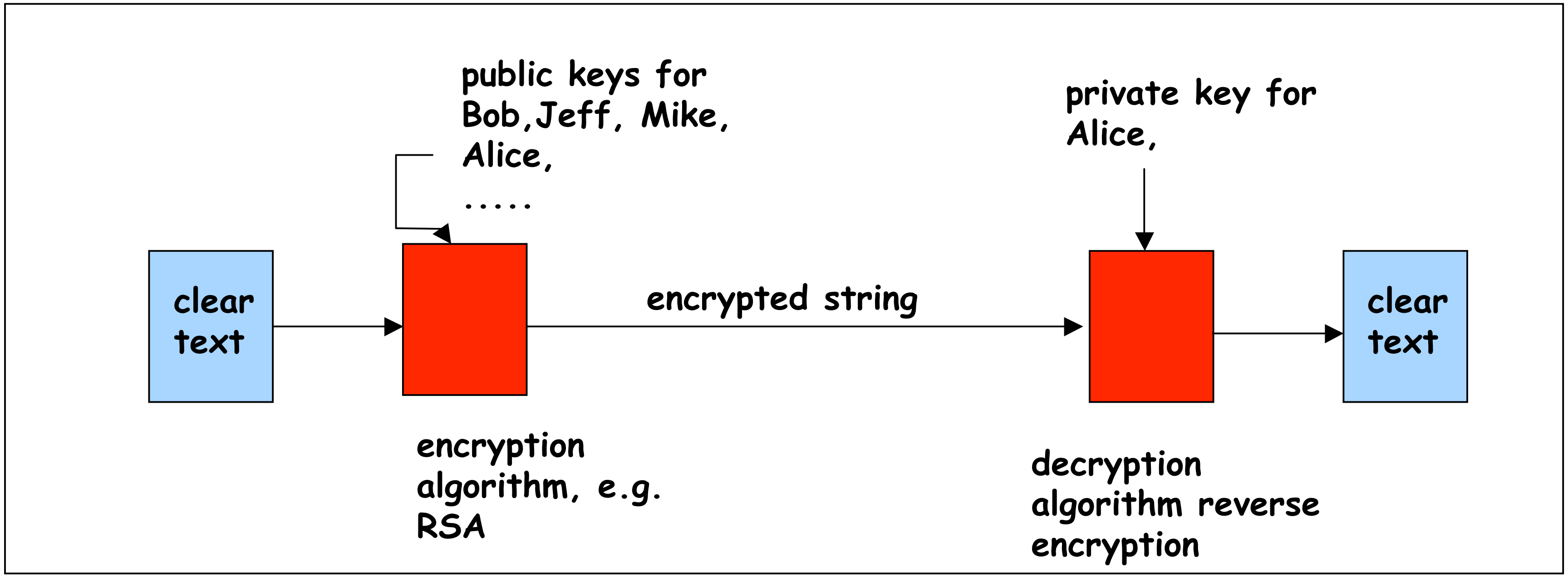

- Receiver calculates the hash value for the document string.
- Receiver applies the public key of the sender E(D(Hash)) to obtain Hash. *
- Then both values are compared and must match.

*Note:it is required that E(D(Hash)) = Hash = D(E(Hash)) !!! This is not true for all encoding functions!

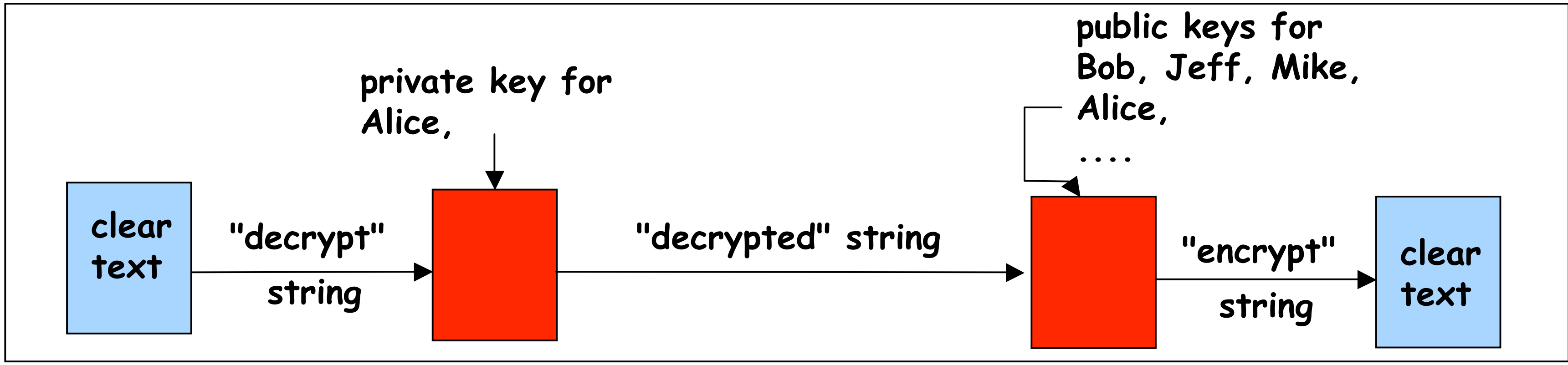
What has to be guaranteed:

1. **Integrity of document:** this can be checked because the document cannot be changed without changing the hash function ("weakly collision" free property)
2. **Authentication of sender:** if the document AND the hash value are changed, then applying the public key of the sender to (D(Hash)) will not deliver a correct result.

# Public key and Digital Signatures




Public Key: document can only be read by Alice




Signature: everyone can read the doc.

http://www.sans.org/dc2000/wileyhacker.pdf

# Attacks to the system

**SANS Institute**
**Washington, D.C. Conference**
**July 7, 2000**


***"Hunting the Wily Hacker'***

May 1988 vol. 31. No. 5 **COMMUNICATION** 484 **OF THE ACM**

## STALKING THE WILY HACKI

*An astronomer-turned-sleuth traces a German trespasser on our military netv through operating system security holes and browsed through sensitive databases. Was*

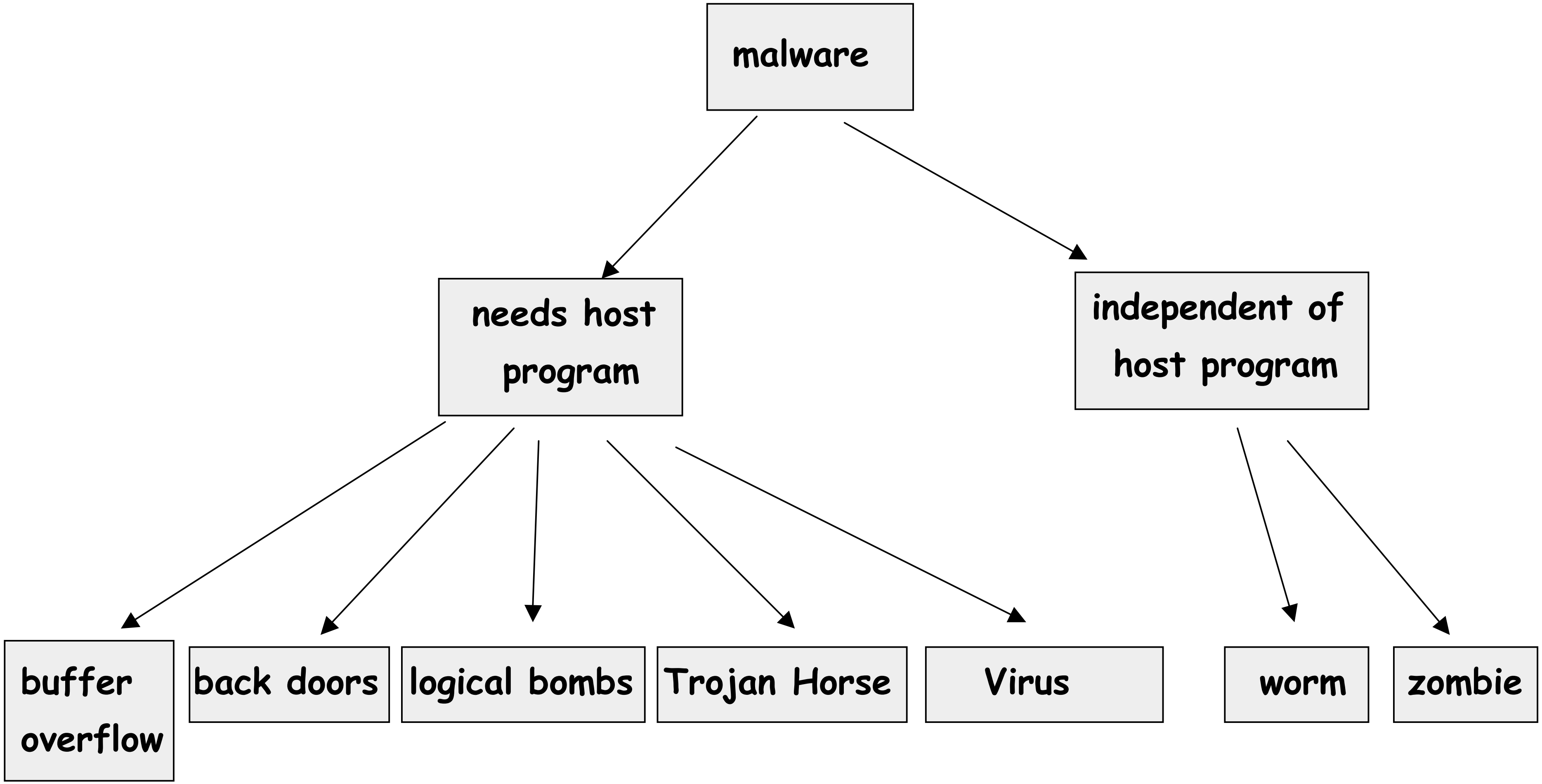
CLIFFORD STOLL

**Craig W. Sorum**
Supervisory Special Agent
FBI Headquarters
202.324.0322
craig.sorum@fbi.gov

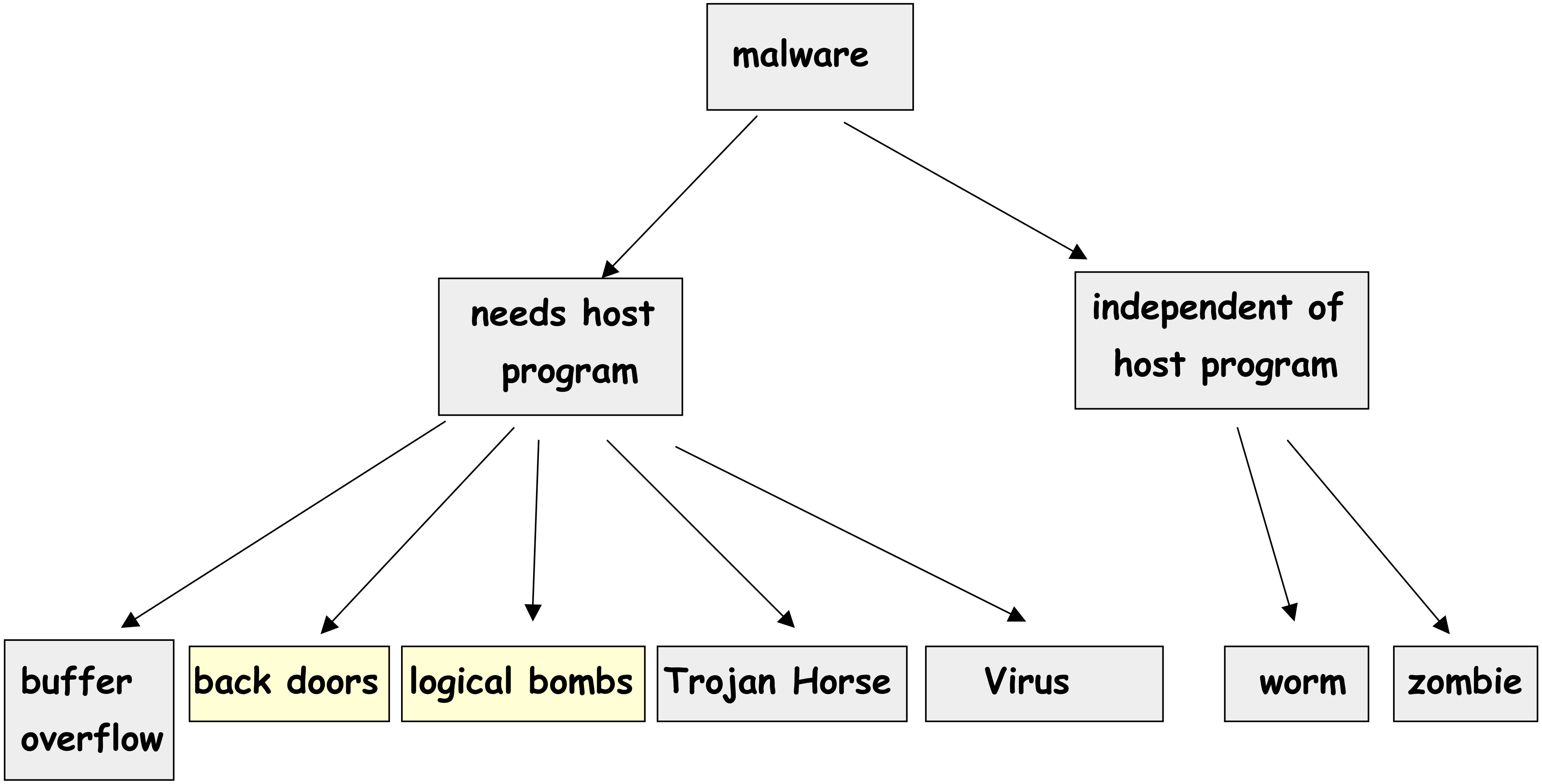
**Gary Harter**
Special Agent
FBI-Washington Field Office
703.762.3024
gharter@leo.gov

***Thank you for your attention!***

# attacks to the system



# attacks to the system



# hidden back doors

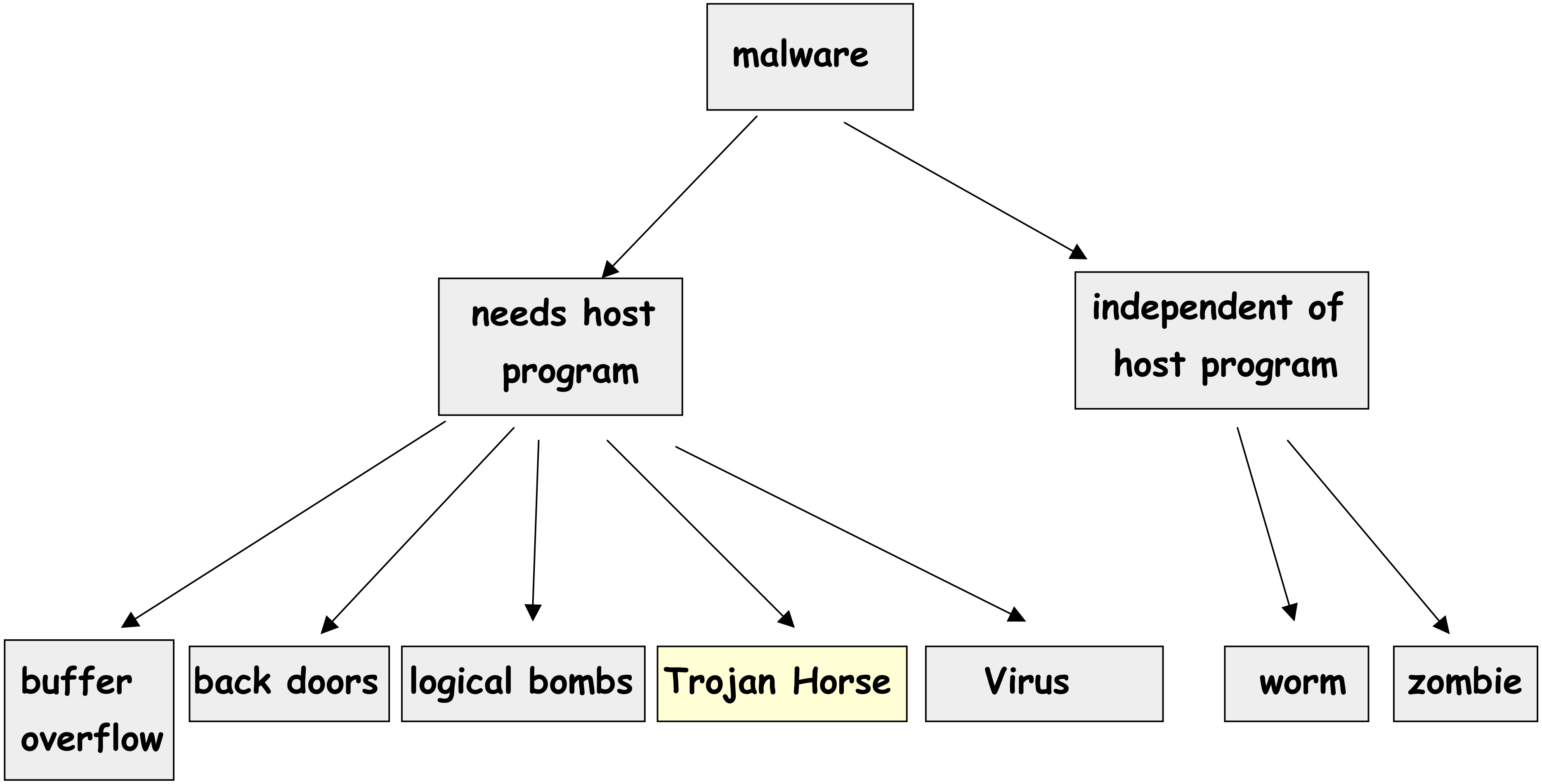
## nomal code

```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing();
        printf("password: ");
        get_string(password);
        enable_echoing();
        v=check_validity(name,password);
        if (v) break;
}
execute shell(name);
```

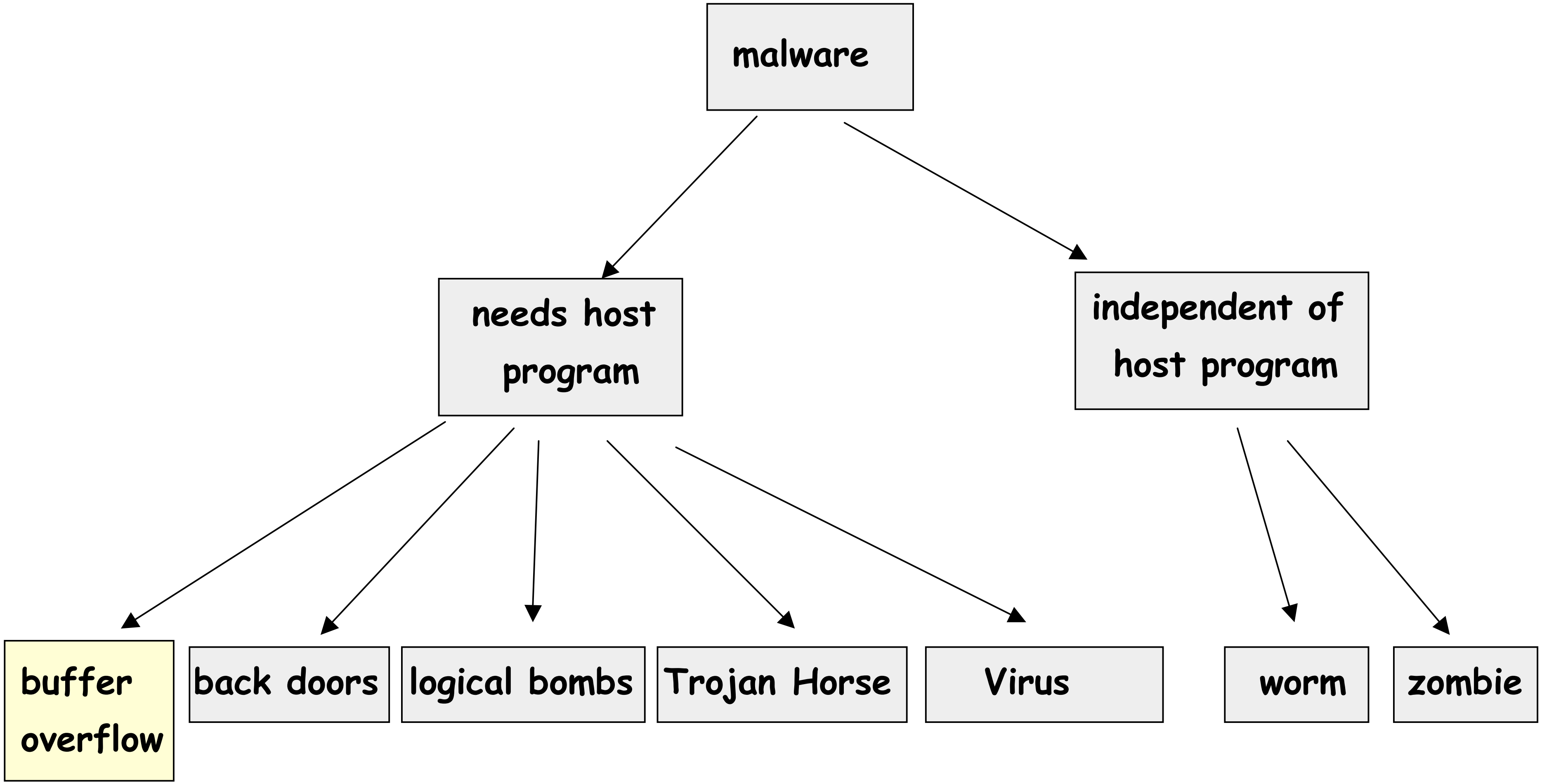
## code with back door

```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing();
        printf("password: ");
        get_string(password);
        enable_echoing();
        v=check_validity(name,password);
        if (v II strcmp(name, "z!5%zy?" == 0) break;
}
execute shell(name);
```

# attacks to the system



# attacks to the system



# buffer overflow

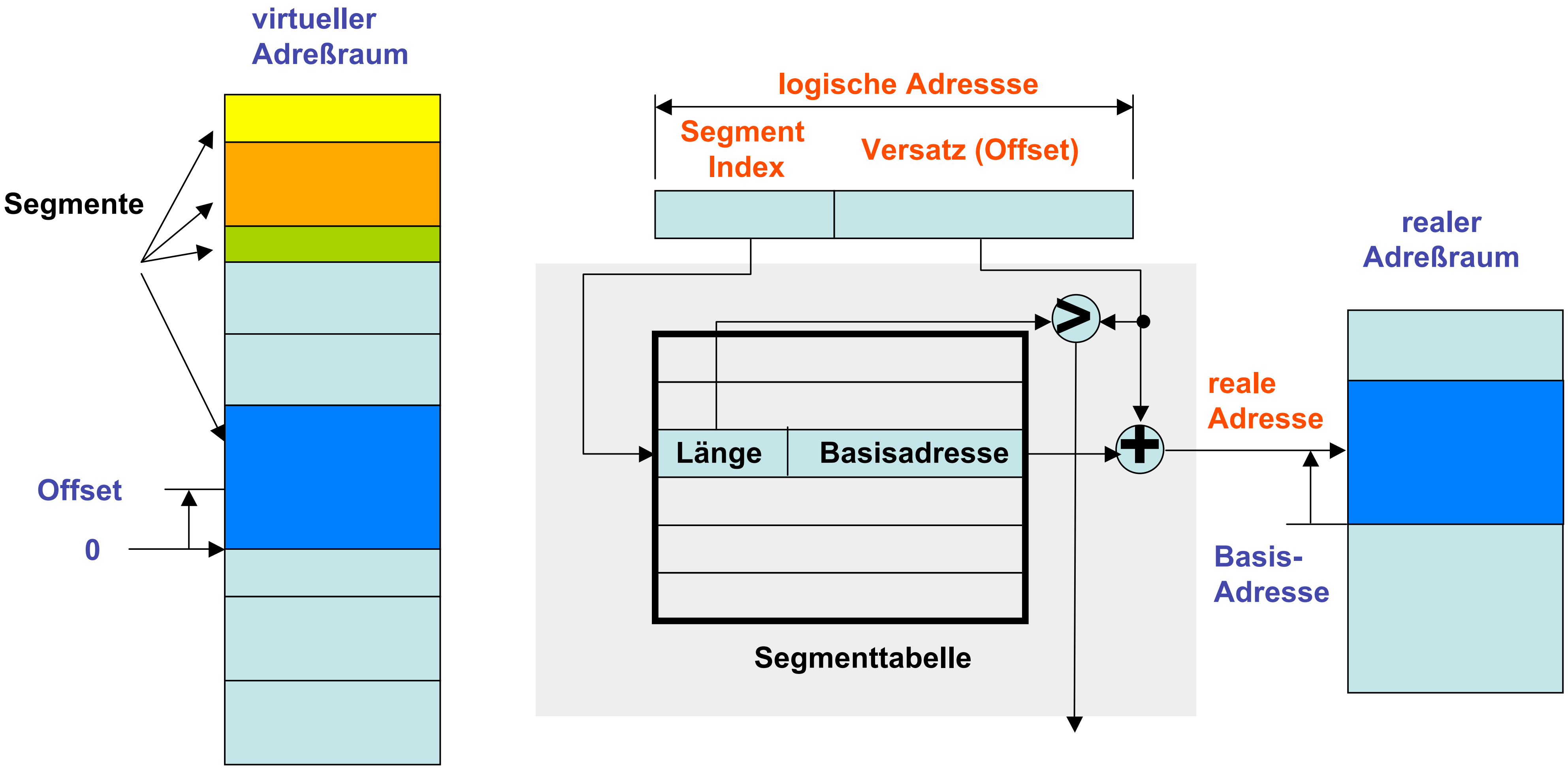
Problem: C-Compiler doesn't check index bounds on arrays.

example:

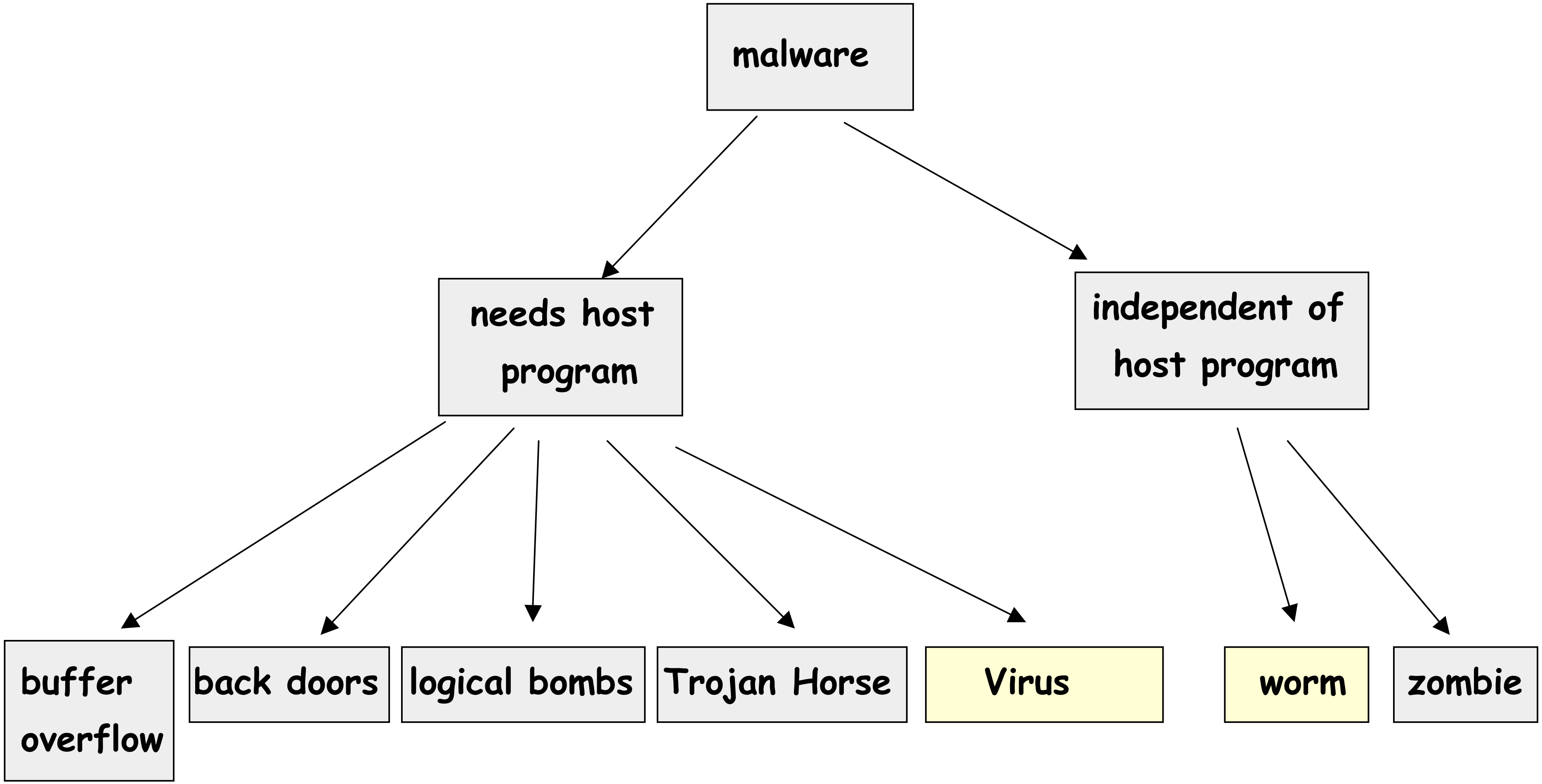
```
int i;
char c[1024];
i=12000;
c[i]=0;
```

Effect: overwrites a byte that is 10976 Bytes away from the index bound.

# "segmentierter" virtueller Speicher



# attacks to the system



# attacks from outside of the system

## Viruses and Worms:

Virus: needs host prgram which is explicitely invoked and executed by a user

Worm: autonomous program which acts completely independent from a user.

Hoax: needs (fooled) user to perform action

- attack over the network
  (or any infected storage device for virus)
- transfer executable code to the victim machine
- often as e-mail attachment (virus)
- replication and distribution by the infected machine

## Geschichte der Computerviren

**1950er** Bell Labs entwickeln ein experimentelles Spiel, in dem die Spieler gegenseitig ihre Computer mit Schäden verursachenden Programmen angreifen.
**1975** John Brunner, Autor von Science-Fiction-Romanen, entwickelt die Idee von einem „Wurm“, der sich in Netzwerken verbreiten kann.
**1984** Fred Cohen führt in einer Dissertation den Begriff „Computervirus“ für Programme mit den entsprechenden Eigenschaften ein.
**1986** Der erste Computervirus, *Brain*, wird angeblich von zwei Brüdern in Pakistan geschrieben.
**1987** Der Wurm *Christmas tree* legt das weltweite IBM-Netzwerk lahm.
**1988** Der *Internet worm* verbreitet sich im US-DARPA-Internet.
**1992** Der *Michelangelo* -Virus sorgt weltweit für Panik, obwohl nur wenige Computer infiziert werden.
**1994** *Good Times*, der erste richtige Virenhoax, erscheint.
**1995** Der erste Dokumentenvirus, *Concept*, erscheint.
**1998** *CIH* oder *Chernobyl* ist der erste Virus, der Computer-Hardware beschädigt.
**1999** *Melissa*, ein Virus der sich selbst per E-Mail weiterleitet, verbreitet sich weltweit. *Bubbleboy*, der erste Virus, der einen Computer allein durch das Lesen einer E-Mail infiziert, erscheint.
**2000** Der *Loveletter-Virus* ist der bisher „erfolgreichste“ Virus. Im selben Jahr tritt der erste Virus für das Palm-Betriebssystem auf, allerdings werden keine Anwender infiziert.
**2001** Ein Virus, der angeblich Bilder der Tennisspielerin Anna Kournikova enthält, infiziert Tausende Computer weltweit.
**2002** David L Smith, Autor von *Melissa*, wird von US-Gerichten zu 20 Monaten Haft verurteilt.
**2003** Der *Blaster*-Wurm verbreitet sich mit Hilfe einer Sicherheitslücke in der Software von Microsoft im Internet. Gemeinsam mit dem E-Mail-Virus *Sobig* macht er den August 2003 zum bisher schlimmsten Monat der Virenvorfälle.
**2004** Die Schöpfer der *Netsky*- und *Bagle*-Würmer wetteifern, welcher Wurm wohl die größeren Auswirkungen hat.

**http://www.sophos.de/sophos/docs/deu/comviru/viru_bde.pdf**

ich hatte die Datei auf der Festplatte und habe sie inzwischen gelöscht!

---Ursprüngliche Nachricht---
From: "a friend"
To: "a friend"
Subject: Achtung Viruswarnung Adressbuch - DRINGEND (fwd)

---Ursprüngliche Nachricht---
From: "Gasthof Alpenhof" <gasthof.alpenhof@rolmail.net>
Habe heute diese Virusmeldung bekommen und den Virus in meiner Datei auch gefunden! Bitte die Anleitung zum Löschen befolgen!

Grüße Renate

> > Ich hoffe, dass Ihr diese Nachricht rechtzeitig erhaltet. Der Virus verbreitet sich von Adressbuch
> > zu Adressbuch, also bitte gleich nachschauen. Er ist in der Tat von
> > Norton und McAfee (und AntiVir 9x) nicht auffindbar. Er schlummert etwa
> > 14 Tage auf dem Rechner, aktiviert sich dann selbst und löscht sämtliche
> > Daten auf der Festplatte.
> >
> > Die Anweisung zu seiner Entfernung ist recht einfach:
> > 1. Auf "Start" klicken, dann auf "Suchen", dann auf Dateien/Ordner
> > 2. In der Suchmaske "sulfnbk.exe"eintippen - so heißt die Virusdatei
> > 3. Bei "Suchen in" muß die Festplatte drin stehen, in der Regen C:
> > 4. Suche starten
> > 5. Wenn diese Datei auftaucht (sie hat ein häßliches schwarzes Icon)
> > - AUF KEINEN FALL ÖFFNEN
> > 6. Mit der rechten Maustaste den Dateinamen anklicken - Löschen
> > drücken
> > 7. Bei der Rückfrage ob die Anwendung tatsächlich in den Papierkorb
> > verschoben werden soll, Ja drücken
> > 8. Auf den Desktop gehen und den Papierkorb öffnen
> > 9. Die Datei "sulfnbk.exe" im Papierkorb suchen und mit der rechten
> > Maustaste löschen
> >
> > Wenn Sie/Ihr die Datei auf Eurem Rechner gefunden habt, sendet diese
E-Mail
> > an alle Kontakte inIhrem/ Eurem Adressbuch, weil der Virus über das
> > Adressbuch verbreitet wird.
> >Danke!

Sorry!!!!

---Ursprüngliche Nachricht--- From: "Dr. S" <--------------->

To: <------------>,"'GK'" <gk>

Subject: "Hoax" (eben kein Virus)

AW: Von Renate - Achtung Viruswarnung Adressbuch - DRINGEND (fwd)

> Ich hatte die Datei auf der Festplatte und habe sie nun gelöscht! .. selbst schuld ..

Bei dieser Nachricht handelt es sich um einen sogenannten Hoax, die Weitergabe der Nachricht ist das Problem, die u.g. Datei ist ein normaler Bestandteil von Windows (z.B. W'98) und dient der Wiederherstellung langer Dateinamen. Wobei ich vermute, dass genug Psychologen in diesem Verteiler sind, die eine derartige sich selbst erfüllende Prophezeiung (die Datei hat wirklich jeder ..) erkennen können ... > > 2. In der Suchmaske "sulfnbk.exe"eintippen - so heißt die Virusdatei

Dr. med. Dipl.-Psych. S,

Zentrum für Telematik im Gesundheitswesen

# What a virus can do:

- **Slow down of E-Mail**. e.g. *Sobig.*
- **Theft of confidential data. e.g.** *Bugbear-D*
- **Website-attacs from YOUR computer**. e.g. *MyDoom*
- **Misuse of YOUR computer by others.**
- **Change of data**. e.g.*Compatable*
- **Deletion of data**. e.g. *Sircam* Wurm
- **Disable hardware**. *CIH* oder *Chernobyl*
- **Jokes**. e.g.*Netsky-D*
- **Display text messages**. e.g. *Cone-F*
- **Loss of credibility**.
- **Embarrasment**. e.g. *PolyPost*

# Virus species

**kind:**

- companion
- overwriting virus
- parasitic virus
- macro virus
- source code virus

**components:**

- user programs
- system programs
- device drivers

**where to hide:**

- "cavities" in the program
- interrupt vector area
- in a memory block marked "used"
- boot sector

**how to hide:**

- stealth virus
- polymorphic virus

# virus actions

sleep until wake-up by some event

start code of virus

search for executable program files

infect program file

- overwrite code with virus code (overwriting virus)
- leave original functionality but add code (parasistic virus)
  - special case "cavity virus".

hide on the computer (memory resident virus)

- hide in the interrupt vector area
- modify bitmap of virtual memory or file system
- hide in the boot sector of the disk (will not be destroyed by formatting)

# Anti-virus techniques

Isolate and identify the virus:

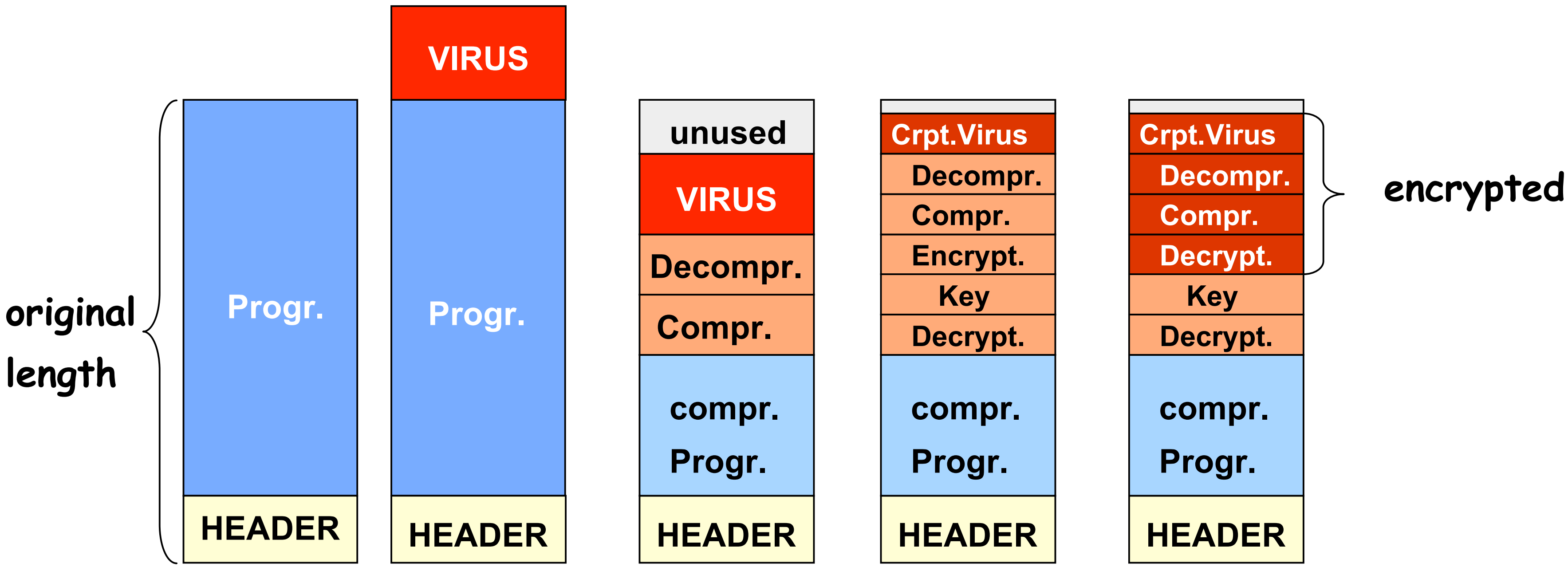
create a protected environment where the impact of a virus can be tested
controlled infection of a specific "goat" file. Goal: Isolation of the virus.
create a listing of the virus code and enter this in a virus database
isolate the code of the virus kernel and create the virus signature

Function of the virus scanner:

compare every file on the disk against the data base of viruses
fuzzy search is required
use signature to identify viruses
use creation date to find modifications since the last check
use length of data to detect infections

# Anti-Anti-virus techniques

- setting creation and modification dates
- exploiting compression and decryption techniques to maintain original length and varying the signature.



# Anti-Anti-virus techniques

## Polymorphic Virus: The many ways to express NOP!

| | | | | |
|---|---|---|---|---|
| MOV A, R1 | MOV A, R1 | MOV A, R1 | MOV A, R1 | MOV A, R1 |
| ADD B, R1 | NOP | ADD #0,R1 | OR R1, R1 | TST R1 |
| ADD C, R1 | ADD B, R1 | ADD B, R1 | ADD B, R1 | ADD C, R1 |
| SUB #4, R1 | NOP | OR R1, R1 | MOV R1, R5 | MOV R1, R5 |
| MOV R1, X | ADD C, R1 | ADD C, R1 | ADD C, R1 | ADD B, R1 |
| | NOP | SHL #0, R1 | SHL #0, R1 | CMP R2, R5 |
| | SUB #4, R1 | SUB #4, R1 | SUB #4, R1 | SUB #4, R1 |
| | NOP | JMP .+1 | ADD R5, R5 | JMP .+1 |
| | MOV R1, X | MOV R1, X | MOV R1, X | MOV R1, X |
| | | | MOV R5, Y | MOV R5, Y |

Sophisticated Viruses comprise a Mutation Engine to perform camouflage automatically.

# Anti and Anti-Anti-virus techniques

The battle goes on:

- How to achieve that an Anti-Virus Program is not infected
- Can access to raw disk help the Virus Scanner?

More techniques which don't help:

- Integrity Checking
- Activity Control

Virus (infection) prevention ?

How to recover from a virus?

Computer Zeitung Nr 11, 10.3.08

Professor Pohlmann: „Trusted Computing bringt Quantensprung"

# Die Virenjagd muss kapitulieren

**Hannover (ab) – Angesichts stetig steigender Trojaner-Fluten geben Antivirenhersteller offen zu: Die traditionelle Schädlingsjagd auf dem PC ist am Ende. Schutzfunktionen sollen künftig verstärkt aus dem Internet kommen. Deutsche Forscher favorisieren dagegen das Trusted Computing.**

**Das Virenproblem** ist mit herkömmlichen Ansätzen nicht mehr zu lösen", mahnt Trend Micros Cheftechnologe Raimund Genes. F-Secure-Chefforscher Mikko Hypponen stimmt zu: „Wir können nicht weitermachen wie bisher." Denn statt gegen einzelne Viren-Freaks kämpft die Security-Branche heute gegen professionell organisierte Cyberkriminelle, die Tag und Nacht Trojaner produzieren.

Die Malware-Flut steigt daher an: Genes erwartet aktuell 1 500 000 neue Schädlinge pro Monat. Und auch wenn die Antivirus-Hersteller ihre Analyse-Infrastrukturen massiv ausbauen und automatisieren, sei diese Flut kaum noch mit Updates auf alle PCs zu bringen.

„Security muss sich ändern: Von einem Produktverkauf zu einem Service", sieht Hypponen künftig die Internetprovider in der Pflicht. Auch Genes will mehr Funktionen „in die Cloud verlagern." Von zentralen Datenbanken will er infektiösen Webseiten schweben ihm ständig aktualisierte Black- und White-Lists mit erlaubten und böswilligen Programmen vor, die die PC-Soft ware dann abfragt.

Professor Norbert Pohlmann vom Gelsenkirchener Institut für Internetsicherheit will dagegen tiefer ansetzen: „Es gibt kein modernes Betriebssystem, das eine angemessene Sicherheit gegen aktuelle Angriffe bietet. Deshalb brauchen wir eine Sicherheitsplattform, die die Basisressourcen der Rechner kontrolliert." Das Opensource-Projekt Turaya nutzt dazu TPM-Kryptochips, Mikrokernel-Betriebssysteme und Virtualisierung samt strenger Prozessisolation. Pohlmann: „Ein Quantensprung für die Sicherheit."

Genes: „Virenproblem ist mit herkömmlichen Ansätzen nicht mehr zu lösen."

# Mobile Code

## Agents, Postscript and Applets

Can we safely execute untrusted code on our computer?

- Sandboxing
- Interpretation
- Digital signatures

# Sandbox

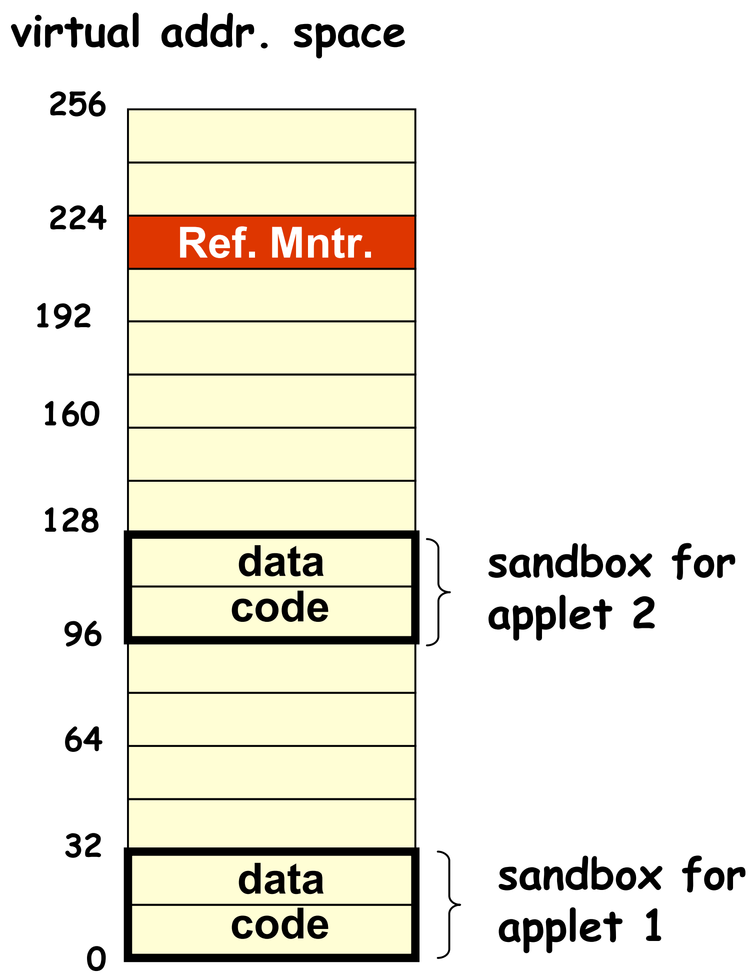
Goal: Separate the virtual address space of a process in areas for trusted and untrusted code.

Problem 1: dynamic jumps where the target address is calculated during run-time.

Solution: Check every "JMP (Rx)" whether its jump target is inside the sandbox.

Problem 2: system calls.

Solution: all system calls are checked by the reference monitor.



## Damage due to poor protection and user ignorance:

## > $10^{12}$ $ / year ??

| | | | | |
|---|---|---|---|---|
| Concept | 1995 | Word Macro | 4 month until widely distr. | $50 Mio. |
| Melissa | 1999 | e-mail W-Macro | 4 days until widely distr. | $385 Mio. |
| Love Letter | 2000 | e-mail Vis.Basic | 5 hours until widely distr. | $15000 Mio. |

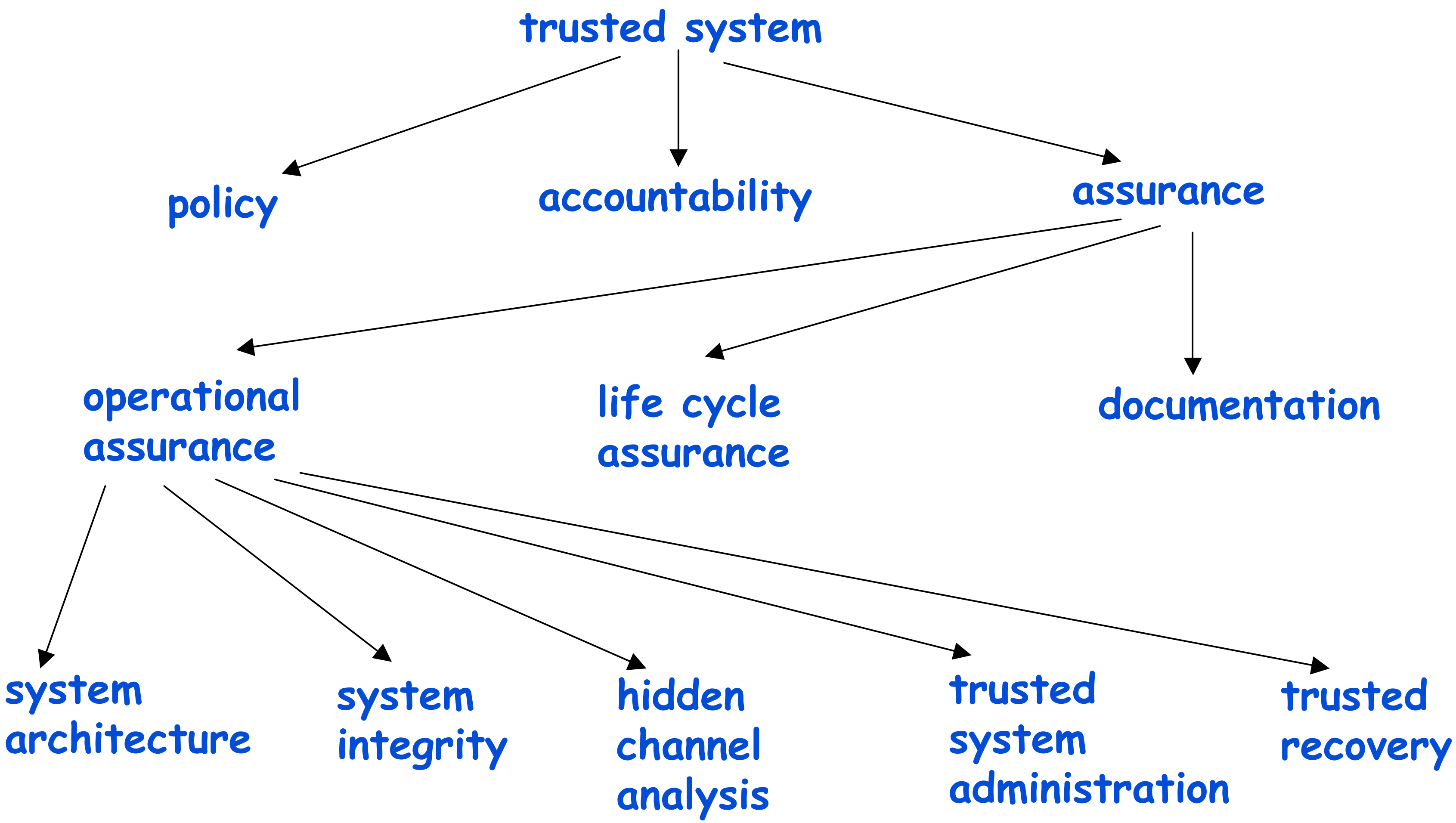
Why not build a trusted and secure computer system ?

Is it possible (with the functionality we are used)?

Is it desirable (or would it be too restrictive) ?

What are the constraints for the user of such a system?

# Requirements for a trusted system DoD Orange Book



| Kriterien | D | C1 | C2 | B1 | B2 | B3 | A1 |
|---|---|---|---|---|---|---|---|
| **Sicherheitsstrategie (Policy)** | | | | | | | |
| Discretionary Access Control (DAC) | | X | X | ⇒ | ⇒ | X | ⇒ |
| Wiederaufbereitung von Objekten | | | X | ⇒ | ⇒ | ⇒ | ⇒ |
| Attribute (Labels) | | | | X | X | ⇒ | ⇒ |
| Attributintegrität | | | | X | ⇒ | ⇒ | ⇒ |
| Export von attributierten Daten | | | | X | ⇒ | ⇒ | ⇒ |
| Attribute für benutzerlesbare Ausgaben | | | | X | ⇒ | ⇒ | ⇒ |
| Mandatory Access Control (MAC) | | | | X | X | ⇒ | ⇒ |
| Subjektattribute | | | | | X | ⇒ | ⇒ |
| Geräteattribute | | | | | X | ⇒ | ⇒ |
| **Abrechenbarkeit** | | | | | | | |
| Identifikation und Authentifikation | | X | X | X | ⇒ | ⇒ | ⇒ |
| Audit-Information | | | X | X | X | X | ⇒ |
| Trusted Path | | | | | X | X | ⇒ |
| **Nachweise** | | | | | | | |
| Systemarchitektur | | X | X | X | X | X | ⇒ |
| Systemintegrität | | X | ⇒ | ⇒ | ⇒ | ⇒ | ⇒ |
| Sicherheitstest | | X | X | X | X | X | X |
| Entwurfsspezifikation und -verifikation | | | | X | X | X | X |
| Analyse verdeckter Kanäle | | | | | X | X | X |
| Trusted Facility-Management | | | | | X | X | ⇒ |
| Konfigurationsmanagement | | | | | X | ⇒ | X |
| Vertrauenswürdige Wiederherstellung | | | | | | X | ⇒ |
| Vertrauenswürdige Verteilung | | | | | | | X |
| **Dokumentation** | | | | | | | |
| Benutzerhandbuch | | X | ⇒ | ⇒ | ⇒ | ⇒ | ⇒ |
| Trusted Facility-Handbuch | | X | X | X | X | X | ⇒ |
| Testdokumentation | | X | ⇒ | ⇒ | X | ⇒ | X |
| Entwurfsdokumentation | | X | ⇒ | X | X | X | X |

Policy

Accountability

Assurance

Documentation

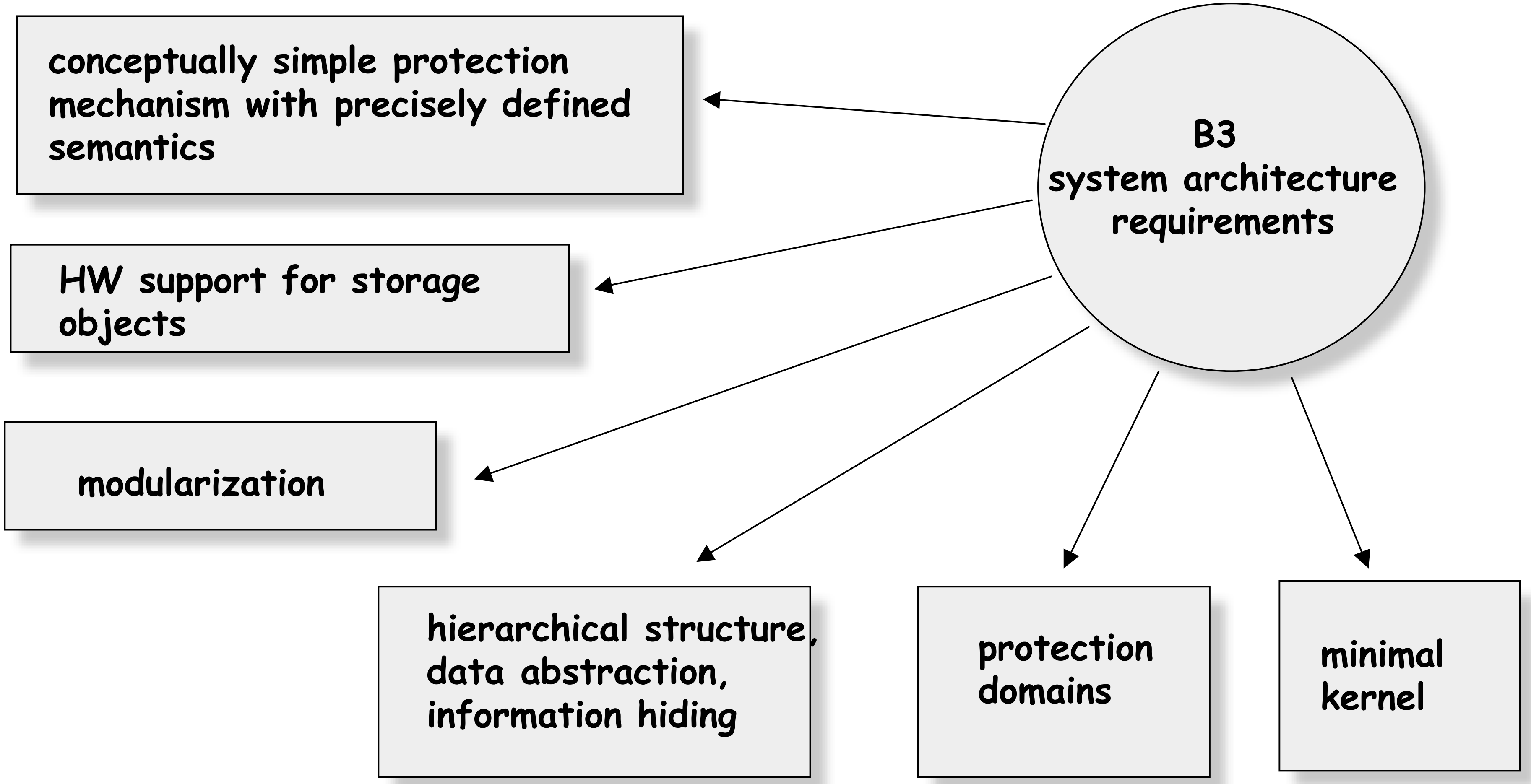
**System Classes acc. to the Orange Book**

X: new requ.

⇒: same requ.

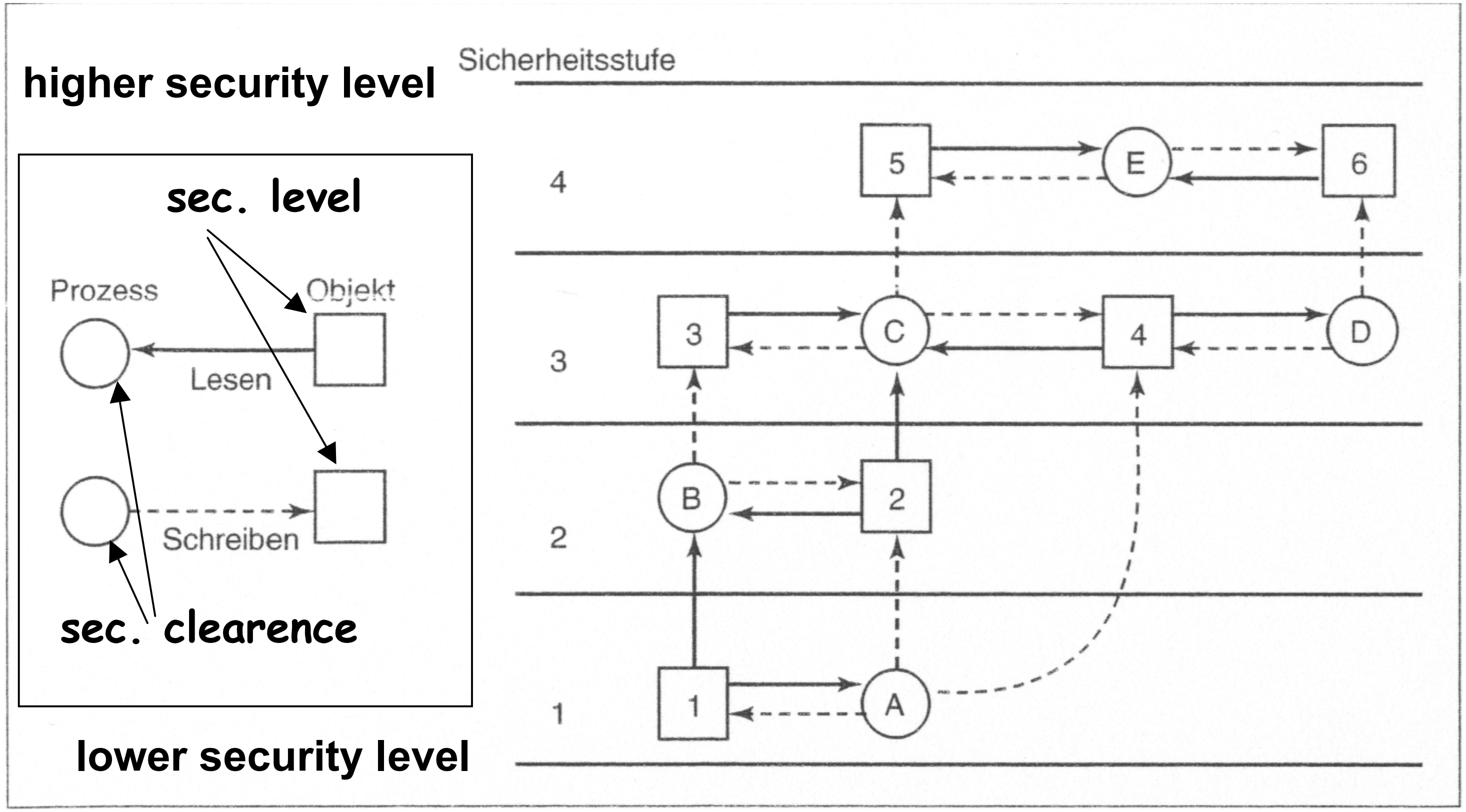
# requirements for the system architecture

acc. to class B3 of DoD Orange Book



# The Trusted Computing Base (TCB)

## Policy: Multi-level security in the Bell-La Padula Model:




simple security property:
a process of security level k can only read objects from security level k-n (n=0,1,2..)

*property:
a process of security level k can only write objects from security level k+n (n=0,1,2..)

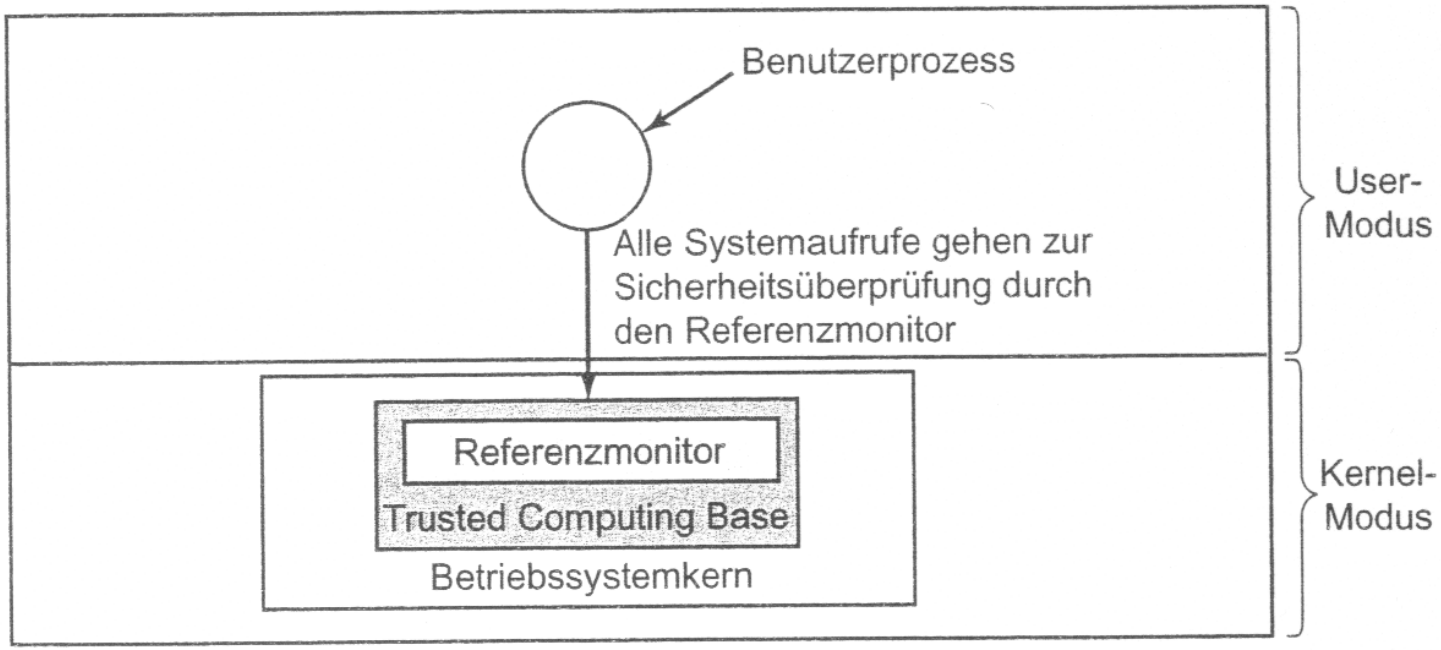
**Objective: Keep a secret !**

# Policy: Multi-level security in the BiBa Model:

simple integrity property: a process of security level k can only write objects from security level k-n (n=0,1,2..)

integrity *rule: a process of security level k can only read objects from security level k+n (n=0,1,2..)

**Objective:** never let a process of lower security clearance overwrite an object of higher security level.


Benutzerprozess
Alle Systemaufrufe gehen zur Sicherheitsüberprüfung durch den Referenzmonitor
User-Modus
Referenzmonitor
Trusted Computing Base
Betriebssystemkern
Kernel-Modus

# design guidelines for secure system

1. Make the design public,
2. Make the default "no access",
3. Check actual rights with every access,
4. Follow the priciple of "least Privilege,
5. The protection mechanism should be simple and the same everywhere in the system
6. The protection mechanism must be acceptable for the user.

**Keep the design simple with orthogonal mechanisms**

## Some references:

Bell&LaPadula (1974) Secure Computer Systems: Mathematical foundations and model, M74-244 MITRE Corp., Bedford, Mass, 1974

Biba (1977) Integrity considerations for secure computer systems, ESD-TR-76-372,ESD/AFSC, Hanscom AFB, Bedford, Mass, April, 1977

DEPARTMENT OF DEFENSE (1985) TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA DOD 5200.28-STD, Supersedes CSC-STD-00l-83, dtd l5 Aug 83Library No. S225,7ll 26.DECEMBER 1985

Clark&Wilson (1987) Comparison of Commercial and Military Computer Security Policies, in Proceedings of the 1987 IEEE Symposium on Security and Pivace, April 27-29, Oakland, Calif, 1987

ITSEC (1991) Information Technology Security Evaluation Criteria (ITSEC) Provisional Harmonised Criteria, June 1991, version 1.2, Luxembourg: Office for Official Publications of the European Communities, 1991

NCSC (1991) Integrity in Automated Informtion Systems, National Computer Security Centre, C Technical Report 79-91, USA September 1991.

# Additial Information

**Sichere Schlüsselaustausch ohne physisches Treffen. Unmöglich??**

**Idee Martin Hellman (1976) (Diffie-Hellman-Merkle-Verfahren).**

**Entsprechung: Schlüssel in Kiste legen. Kiste mit einem Schloss versehen, an Adressaten schicken. Adressat bringt weiteres Schloss an und schickt die Kiste zurück. Eigenes Schloss wird entfernt und Kiste wieder an Adressaten. Der kann nun sein Schloss entfernen, die Kiste öffnen und den Schlüssel entnehmen. Folgende Kisten müssen nur noch mit dem entsprechenden Schloss gesichert sein.**

**Sicheres vereinbaren von Schlüsseln- kein Austausch einen Geheimnisses. Dazu wird z.B. DES benutzt.**

**Problem: Vereinbarung von Schlüsseln erfolgt synchron mit einem konkreten Partner.**
**umständlich, mehrere Nachrichten müssen ausgetauscht werden, Partner muss gemäß des Protokolls antworten.**
**Symmetrischer Schlüssel, d.h. ver- und entschlüsseln wird mit demselben Schlüssel durchgeführt.**

**Public-Key Verfahren:**

**Idee: Whitfield Diffie**

**Asymmetrischer Schlüssel. Ver- und Entschlüsselung mit unterschiedlichen Schlüsseln.**

**Öffentlicher Schlüssel zum Verschlüsseln, Privater Schlüssel zum Entschlüsseln.**

**Entsprechung: Jeder der eine Nachricht an A bekommt eine Menge von Schnappschlössern. Nur A hat den entprechenden Schlüssel. Wenn das Schnappschloss eingerastet ist, kann nur A die Kiste öffnen.**

**Gesucht: eine Einwegfunktion, die eine solche Asymmetrie unterstützt. Sie muss sich z.B leicht umkehren lassen (Falltürfunktion) z.B. im Gegensatz zu Passwd-Verschlüsselung.**

**Erste Veröffentlichung der Idee: 1975**

**Rivest 1977 hat die Idee. Unter RSA veröffentlicht (Ronald Rivest, Adi Shamir,Leonard Adleman)**

**Verschlüsselung:** $C = K^e \pmod{(p \cdot q)}$ **Alice ist bekannt sind: p, q, e und K**

**Öffentlich sind:** $N = p \cdot q$ **und e**

**Entschlüsselung:** $K = C^d \pmod{(p \cdot q)}$

**d berechnet sich aus e,p,q mit:** $d \cdot e = 1 \pmod{\varphi N} = 1 ((\bmod ((p-1) \cdot (q-1)))$

**Sicherheit:**

**200-stelliges N --> auf 80 Rechnern 2003 -2005**

**Kommerziell: 300 Stellen**

**Problem: Ver- und Entschlüsselung sind aufwändig.**

**RSA wir in der Regel in Hybridvefahren (Kombination mit sym. Verschl.) angewendet:**

**zufälliger Sitzungschlüssel wird generiert per RSA verschlüsselt und mit der NAchricht übertragen??**

**Digitale Signaturen:**

**Eigenschaften:**

| | |
|---|---|
| **authentisch:** | **der Erzeuger muss identifizierbar sein** |
| **fälschungssicher:** | **sowohl der Erzeuger als auch ein Angreifer kann die Signatur nicht beliebig verändern.** |
| **unwiderrufbar:** | **Der Erzeuger kann die Unterschrift nicht ungültig machen.** |