LEHRSTUHL 14
SOFTWARE ENGINEERING

Vorlesung

Methodische Grundlagen des Software-Engineering

im Sommersemester 2013

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

3.2: Sicherheitsanforderungen

v. 13.06.2013

3.2 Sicherheitsanforderungen

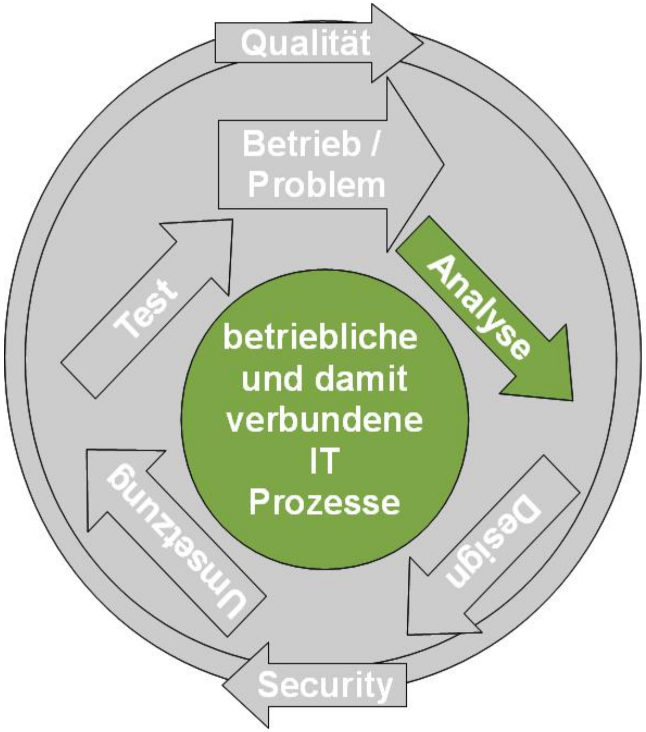
**[inkl. Beiträge von
Prof. Dr. Joachim Biskup (TU Dortmund)]**

Literatur:

[Bis09] Joachim Biskup: **Security in Computing Systems - Challenges, Approaches and Solutions**, Springer-Verlag 2009.
Unibibliothek (e-Book): http://www.ub.tu-dortmund.de/katalog/titel/1350771
Papier-Version: http://www.ub.tu-dortmund.de/katalog/titel/1231626
[Jür05] Jan Jürjens: **Secure systems development with UML**, Springer-Verlag 2005.
Unibibliothek (e-Book): http://www.ub.tu-dortmund.de/katalog/titel/1361890
Papier-Version: http://www.ub.tu-dortmund.de/katalog/titel/1091324

Einordnung
3.1 Sicherheitsanforderungen

- Geschäftsprozessmodellierung
- Process-Mining
- Modellbasierte Entwicklung sicherer Software
  - Model-Driven Architecture
  - Sicherheitsanforderungen
  - UMLsec
  - UML-Analysis
  - Design Principles
  - Examples
    - TLS Variant
    - CEPS Purchase



Security Requirements

Aspects

Protection of the system, against attacks	Protection of environment, against faults
Security	**Safety**

Basic Goals

Integrity	Confiden-tiality	Availability	Account-ability	Nonrepu-diability	Stability, Reability, Robustness, Maintainability, Plausibility, Correctness, Trustability, ...

Basic Functions

Identification / Authentication	Authorization / Rights managem.	Rights control	Logging	Fault tolerance	Control	...

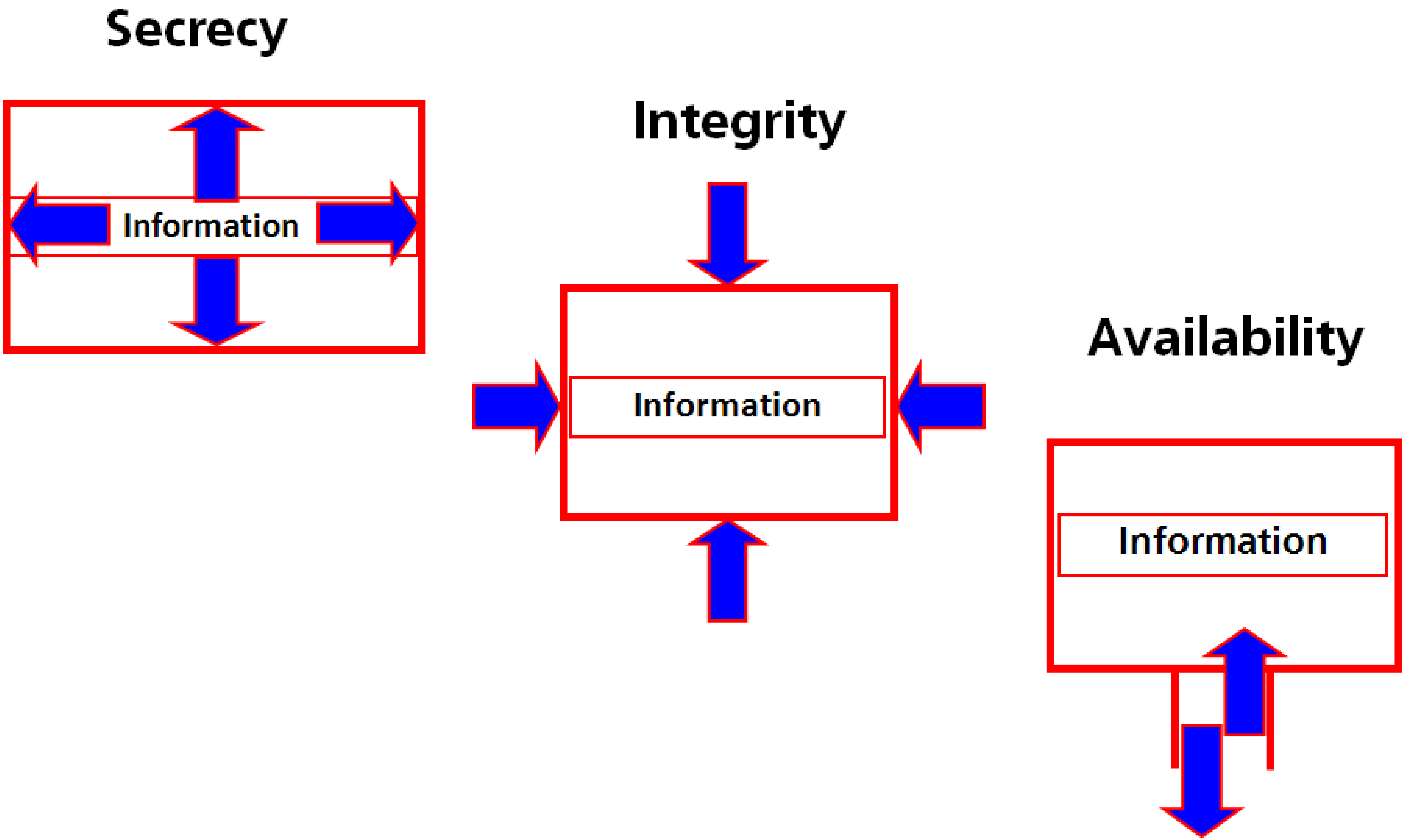
Mechanisms

Chip cards / Passwords	Separation of duty	Access Control: discrete / global	Crypto-graphy	Communic. protocols	Audit Logs	Redundancy	...

Security interests: an expanded list

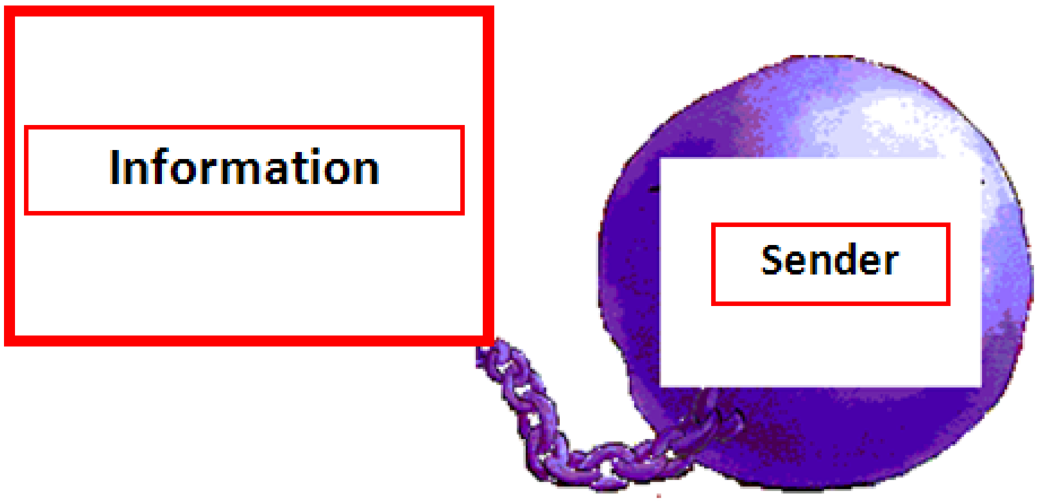
- availability
- integrity: correct content
- integrity: unmodified state
- integrity: detection of modification
- authenticity
- non-repudiation
- confidentiality
- non-observability
- anonymity
- accountability
- evidence
- integrity: temporal correctness
- separation of roles
- covert obligations
- fair exchange
- monitoring and eavesdropping

Basic Security Requirements I

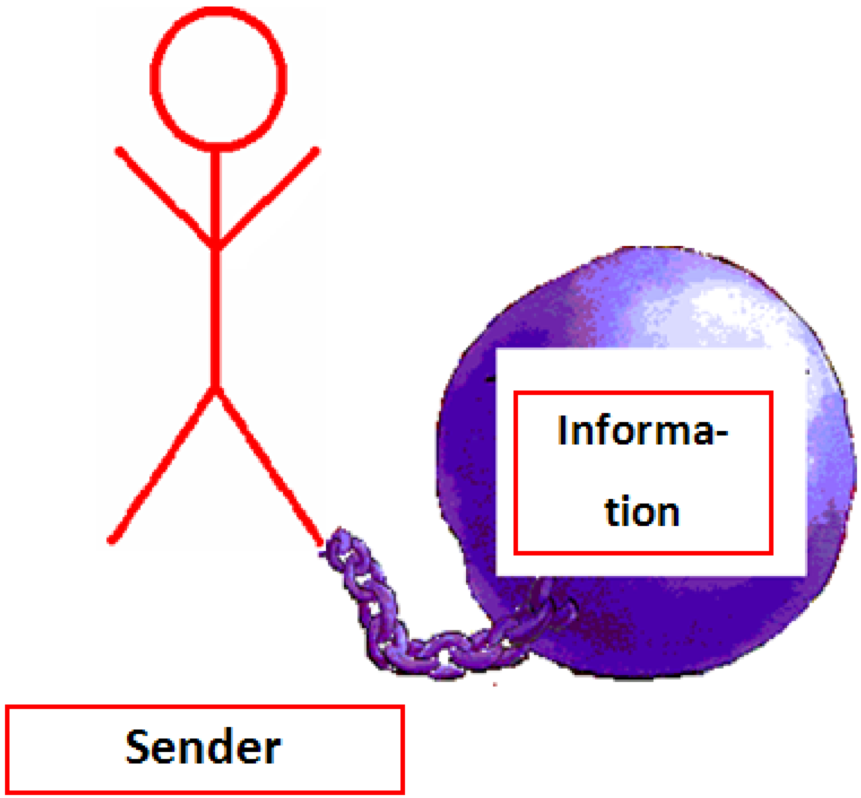


Basic Security Requirements II

Authenticity



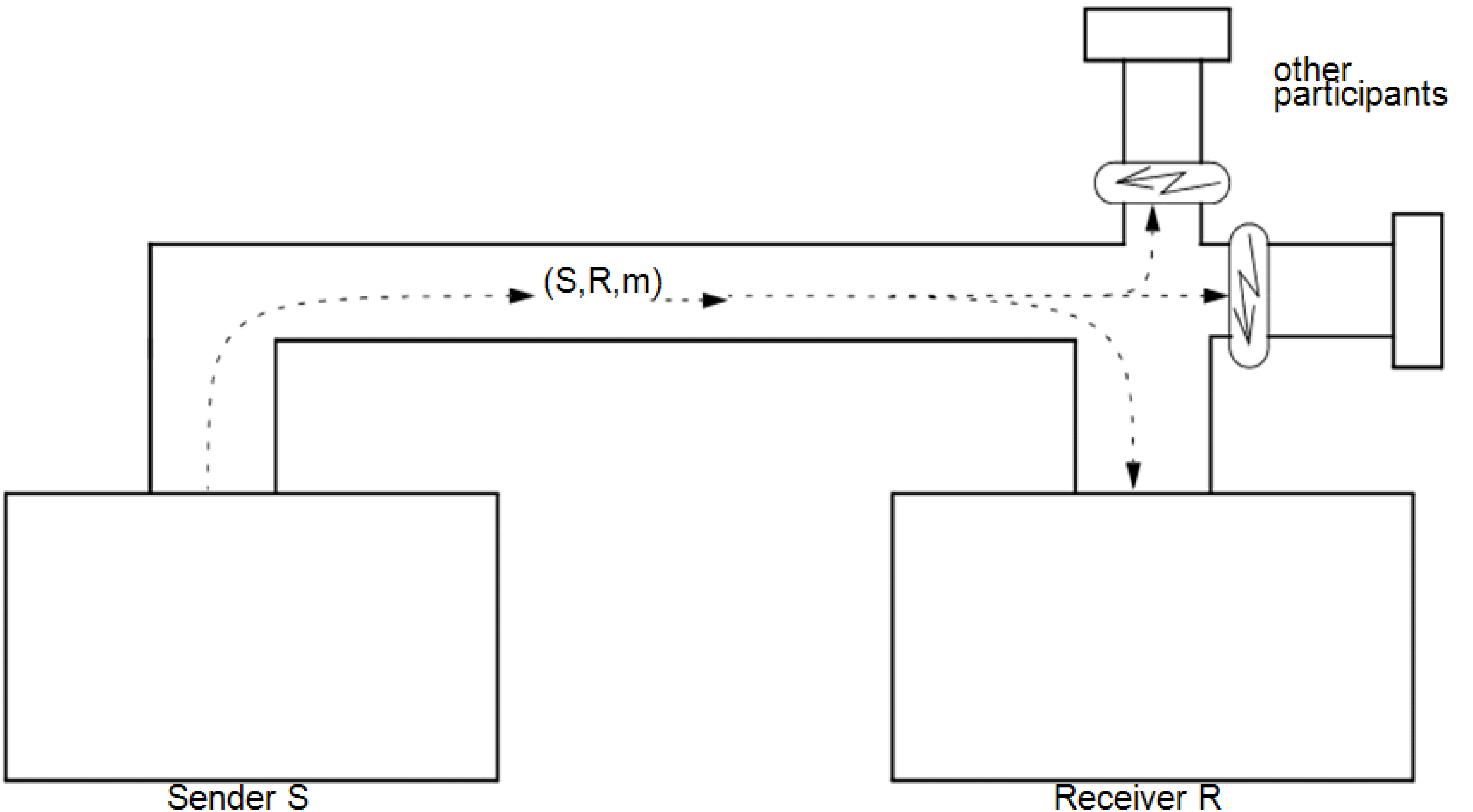
Nonrepudiability



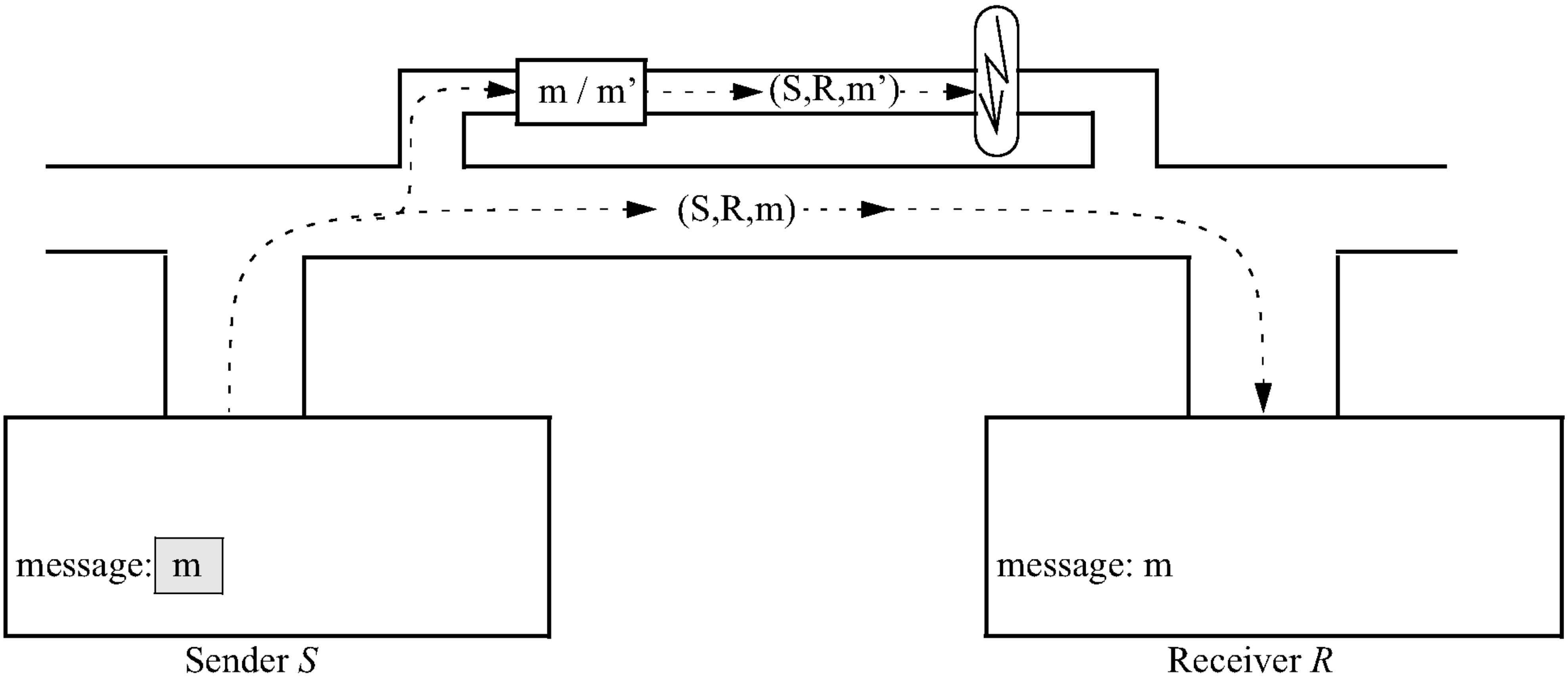
Security requirements: Secrecy and Integrity

- Two of the main data security requirements are **secrecy** (or **confidentiality**) and **integrity**.
- Secrecy of data means that the data should be **read** only by legitimate parties.
- Integrity of data means that it should be **modified** only by legitimate parties.

Confidentiality



Integrity: unmodified state



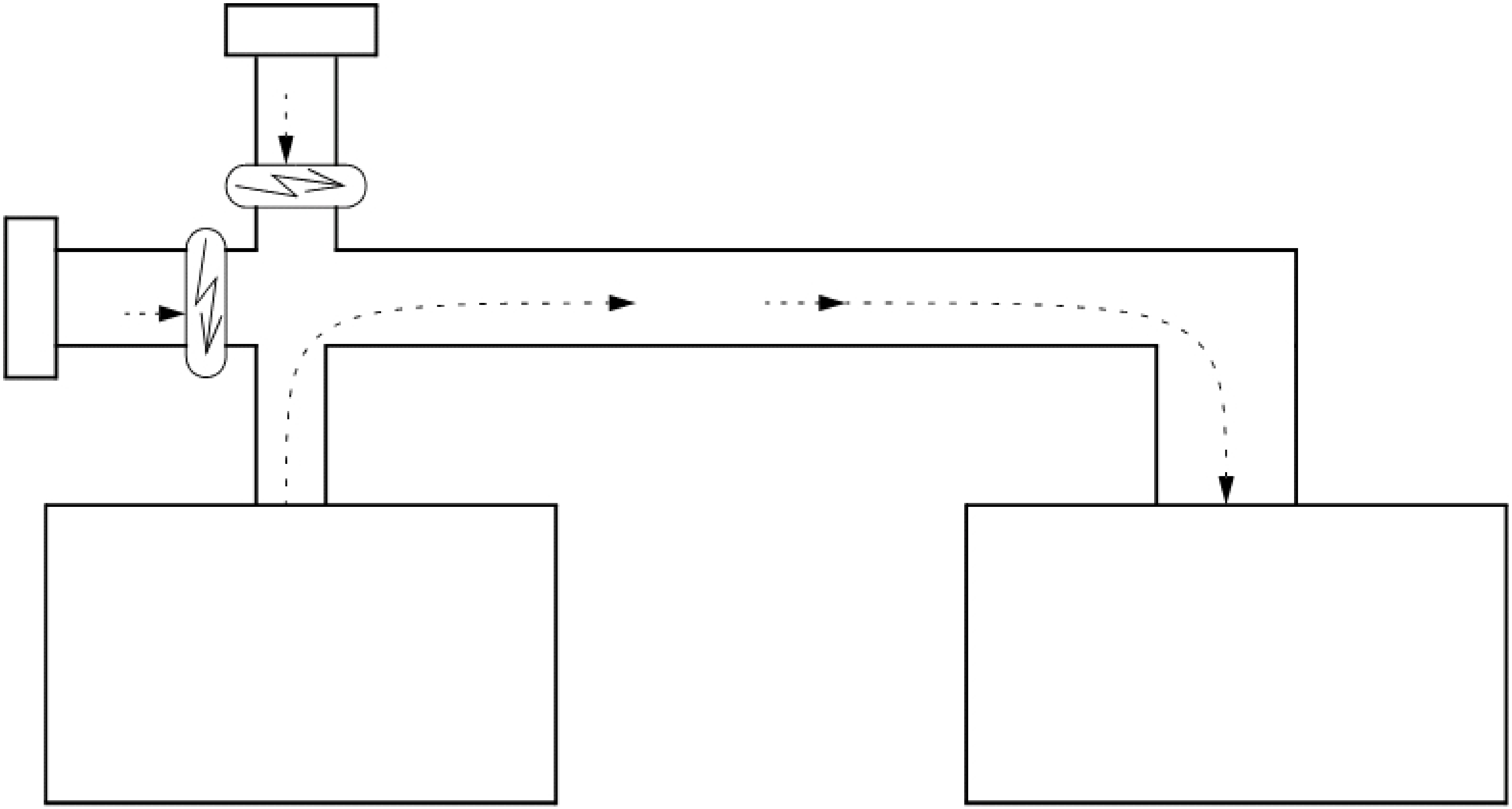
Security requirements: Secure Communication Link

- Sensitive communication between different parts of a system needs to be protected.
- The relevant requirement of a **secure communication link** is here assumed to preserve secrecy and integrity for the data in transit.

Security requirements: Authenticity

- There are different variants of this third main security requirement.
- Two important ones are message authenticity and entity authenticity.
- **Message authenticity** (or **data origin authenticity**) means that one can trace back some piece of data to what its original source was, at some point in the past.
- **Entity authenticity** ensures that one can identify a participant in a protocol, and in particular make sure that the party has actually actively participated in the protocol at the time.
- The process providing authenticity is called **authentication**.

Authenticity



Security requirements: Non-repudiation

- One way of providing fair exchange is by using the security requirement of **non-repudiation** of some action, which means that this action cannot subsequently be successfully denied.
- That is, the action is **provable**, usually with respect to some trusted third party.

Security requirements: Fair Exchange

- When trading goods electronically, the **fair exchange** requirement postulates that the trade is performed in a way that prevents the participating parties from cheating.
- If for example buyer has to make a prepayment, the buyer should be able to prove having made the payment and to reclaim the money if that good is subsequently not delivered.

Fundamental aspects of security

- *security* is a comprehensive property
- *security* design reflects the *interests* of *participants*
- *conflicts* must be balanced
- *security requirements* identify *informational activities* and their *threats*
- *security mechanisms* aim at
  - *preventing* security violations
  - *limiting* the damage caused by violations
  - *compensating* their consequences

Security evaluation

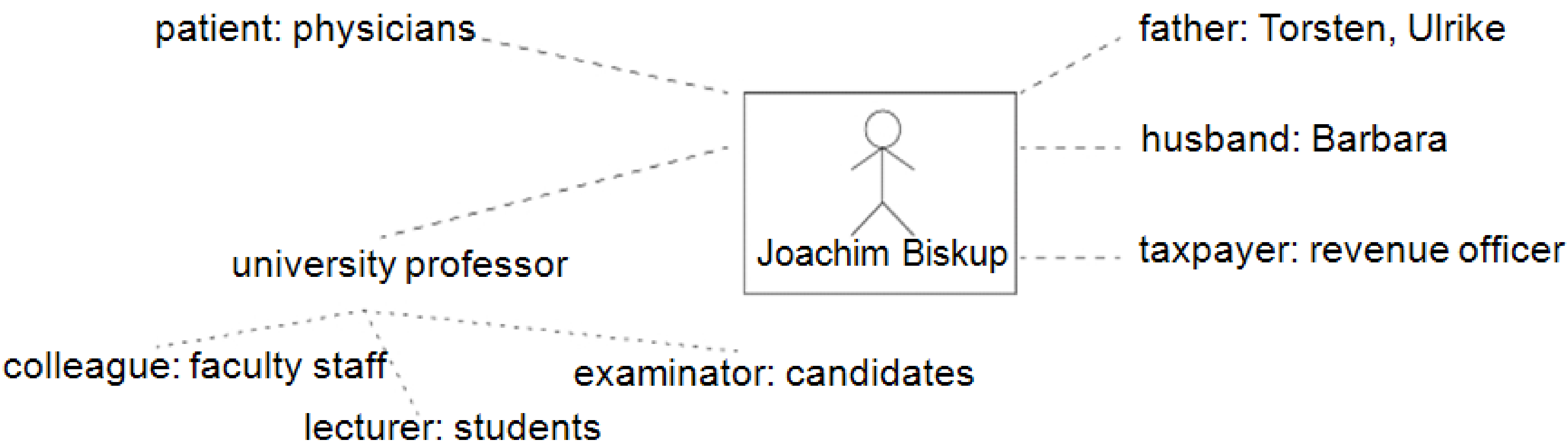
- whether, or to what extent,
  do security mechanisms *satisfy* the security requirements?
- which *assumptions* are underlying the evaluation?
- which kind of *trust* is assigned to participants or system components?
- do the *risks* recognized justify
  the *expenditure* for the security mechanisms selected?

Requirements by legislation: important examples

- *privacy acts detailing the principles of informational self-determination*
  first declare a general and protecting forbiddance,
  and then allow the processing of personal data under specific conditions
- *telecommunication and services acts*
  enable the public and commercial exploitation of informational activities, and
  lay foundations for legally binding transactions
  in public administration and private commerce
- *intellectual property acts*
  support and extend the traditional concept of
  authors' (or their publishers') copyright in texts or images
  to all kinds of electronic multimedia objects
- *criminal acts* (laws)
  identify definitely offending behavior within computing systems

Privacy and informational self-determination

- an individual determines by himself
  which personal *information* he is willing to *share* with group members
  in a specific social *role*



- *an individual selects his social roles under his own responsibility*
- *other agents respect the intended separation of roles,*
  refraining from unauthorized information flows between different roles

Protection rules for personal data

- *based on permission*:
  personal data should be processed only by permission,
  expressed in a law or with the explicit consent of the person concerned
- *need-to-know*:
  processing personal data should be restricted to actual needs,
  preferably by avoiding the collection of personal data at all or
  by converting it into non-personal data by anonymization
- *collected from the source*:
  personal data should be collected from the person concerned
- *bound to original purpose*:
  personal data should be processed only for the well-defined purpose
  for which it was originally collected

Protection rules for personal data

LEHRSTUHL 14
SOFTWARE ENGINEERING

- *subject to inspection*:
  a person concerned should be informed about the kind of processing that employs his personal data
- *under ongoing control*:
  "wrong" personal data should be corrected;
  "no longer needed" personal data should be deleted
- *with active support*:
  agents processing personal data are obliged to actively pursue the privacy of the persons concerned

Requirements by security evaluation criteria

- *Trusted Computer System Evaluation Criteria* (TCSEC),
  known as the *Orange Book*,
  issued by the US Department of Defense

- *Information Technology Security Evaluation Criteria* (ITSEC),
  jointly published by some European countries

- *Common Criteria for Information Technology Security Evaluation* (CC),
  a version of which has also become an *ISO standard*

Common Criteria: security functionality

- *Audit*, as the basis of monitoring and analyzing the behavior of participants
- *Communication*, with an emphasis on providing evidence for sending and receiving of messages
- *User Data Protection*, with an emphasis on enforcing availability, integrity and confidentiality of the users' objects
- *Identification and Authentication*, for enforcing authenticity with non-repudiation and accountability
- *Privacy*, including: non-observability, anonymity, pseudonymity and unlinkability
- *Protection of the Trusted Security Functions*, which deals with the installation, administration and operation of security mechanisms, i.e., how security mechanisms are securely protected in turn
- *Resource Utilization*, including fault tolerance, priorization and scheduling
- *Target of Evaluation Access*, including log-in procedures
- *Trusted Path / Channel*, dealing with the physical link between a (human) participant and the (processor of the) technical device employed

Common Criteria: evaluation assurance levels

- EAL1: functionally tested
- EAL2: structurally tested
- EAL3: methodically tested and checked
- EAL4: methodically designed, tested and reviewed
- EAL5: semi-formally designed and tested
- EAL6: semi-formally verified design and tested
- EAL7: formally verified design and tested

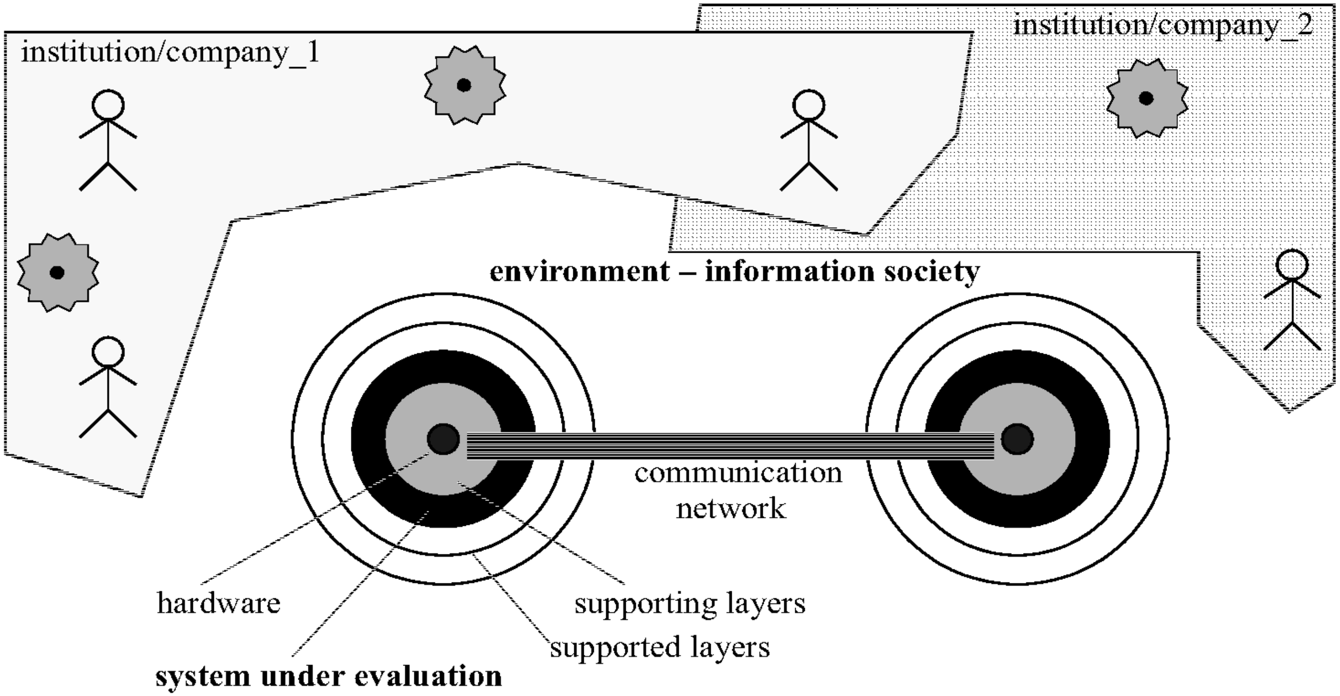
Common Criteria: top-level assurance classes

- Configuration Management
- Delivery and Operation
- Development
- Guidance Documents
- Life Cycle Support
- Tests
- Vulnerabilities

for each of the subclasses of the assurance classes, appropriate assurance levels are required

A practical checklist for evaluations

- a comprehensive view of the circumstances



- answers to the following questions:
- on what other *components*, in what layers, is the system based?
- in what *environment* is the system embedded?
- in what *institution* or *company* is the system used?

Issues for the actual version, configuration and circumstances

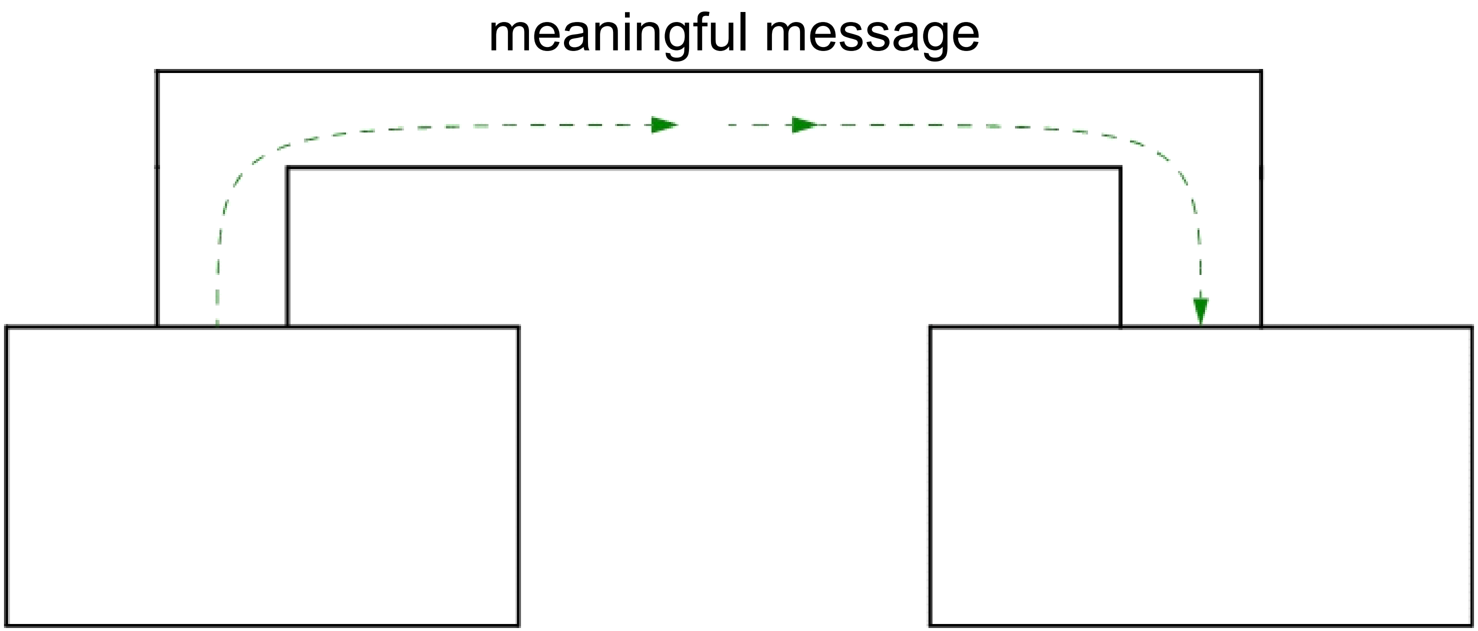
- ***security policy:***
  are the security requirements explicitly expressed?
- ***authorization:***
  is every access (execution of an operation by a subject on an object), preceded by an explicit permission
  (granting a corresponding access right/a suitable cryptographic key)?
- ***control:***
  is such a permission controlled before execution,
  (by checking access rights/by the need for a suitable cryptographic key)?
- ***authenticity:***
  is the authenticity of all items checked before the execution?
- ***monitoring:***
  can intrusions be detected, though potentially only afterwards, and
  can any resulting damage be limited or compensated?
- ***total coverage:***
  do the security mechanisms cover all accesses and messages?

Construction principles

- ***open design:***
  the design and the actual implementation of security mechanisms may or even must be made public (“no security by obscurity”)
- ***fail-safe defaults:***
  any informational activity within a computing system is forbidden unless it has been explicitly permitted
- ***fine granularity:***
  elementary, independent activity classes are defined as units of control
- ***need-to-know / need-to-act:***
  permissions are granted only if they are strictly needed
- ***complete mediation:***
  permissions are granted to well-defined *single* activity executions
- ***economy of mechanisms:***
  the main burden of security enforcement is put on technical mechanisms
- ***complexity reduction:***
  the security mechanisms are appropriately concentrated

[Saltzer, Schroeder: The Protection of Information in Computer Systems, Communications of the ACM 17, 7, 1974]

Message transmission: basic abstraction for challenges

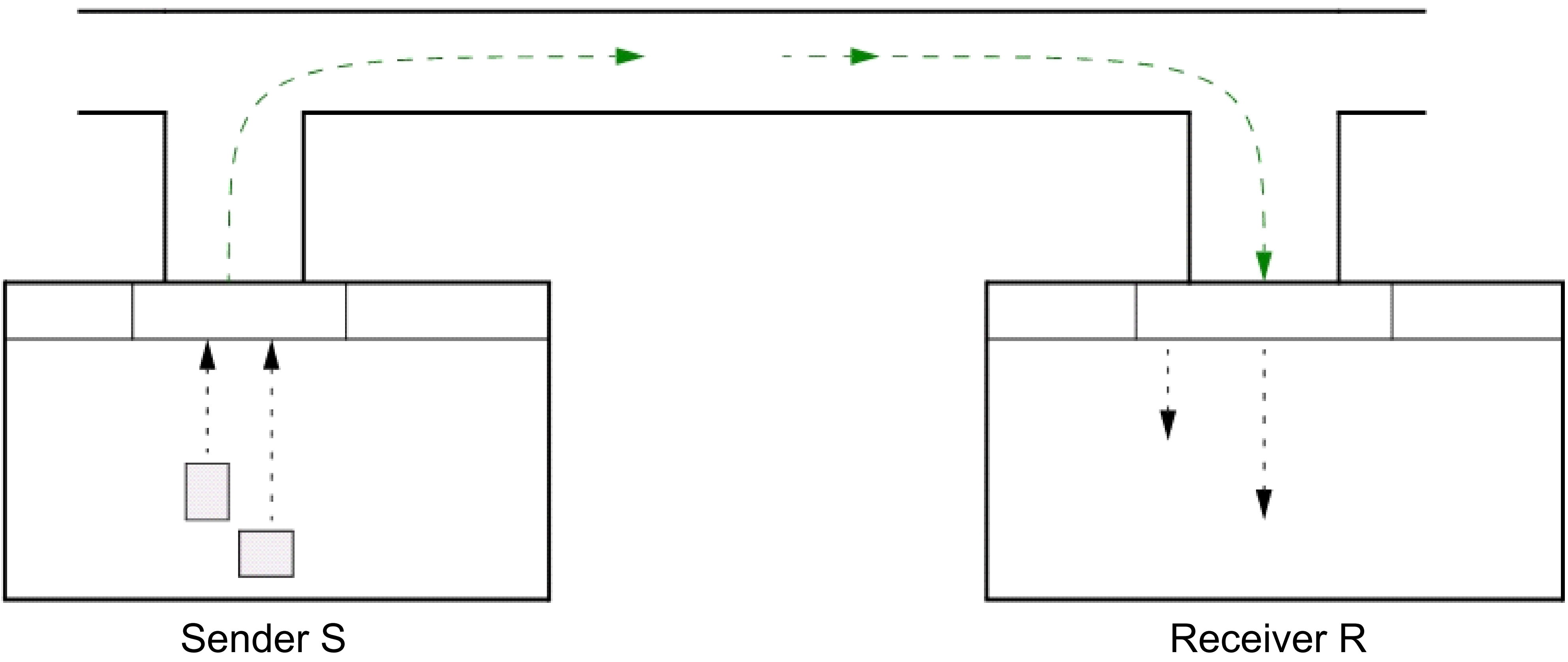


- captured by an *assignment* statement of the form R:=S
- the content *m* of the memory part denoted by *S*
  is transmitted to the memory part denoted by *R*
- *S writes* into *R*, or
  R reads from *S*, or
  some mechanism *pushes* the transmission

Transmission control in distributed computing systems: example

sender::send_data(receiver,message)

receiver::receive_data(sender,message)



Security requirements: Secure Information Flow

- A traditional way of ensuring security in computer systems is to design **multi-level secure** systems.
- In such systems, there are different levels of sensitivity of data.
- For simplicity, one usually considers two security levels: **high**, meaning highly sensitive or highly trusted, and **low**, meaning less sensitive or less trusted.
- Where trusted parts of a system interact with untrusted parts, one has to ensure that there is no indirect leakage of sensitive information from a trusted to an untrusted part.

Security requirements: Secure Information Flow

- To ensure secure information flow, one enforces the “no down-flow” policy: **low** data may influence **high** data, but not vc. vs..
- The opposite of this condition, “no up-flow”, enforces that untrusted parts of a system may not indirectly manipulate high data: **high** data may influence **low** data, but not vc. vs..
- These security requirements, called **secure information flow** or **non-interference** are rather stringent definitions of secrecy and integrity which can detect implicit flows of information that are called **covert channels**.

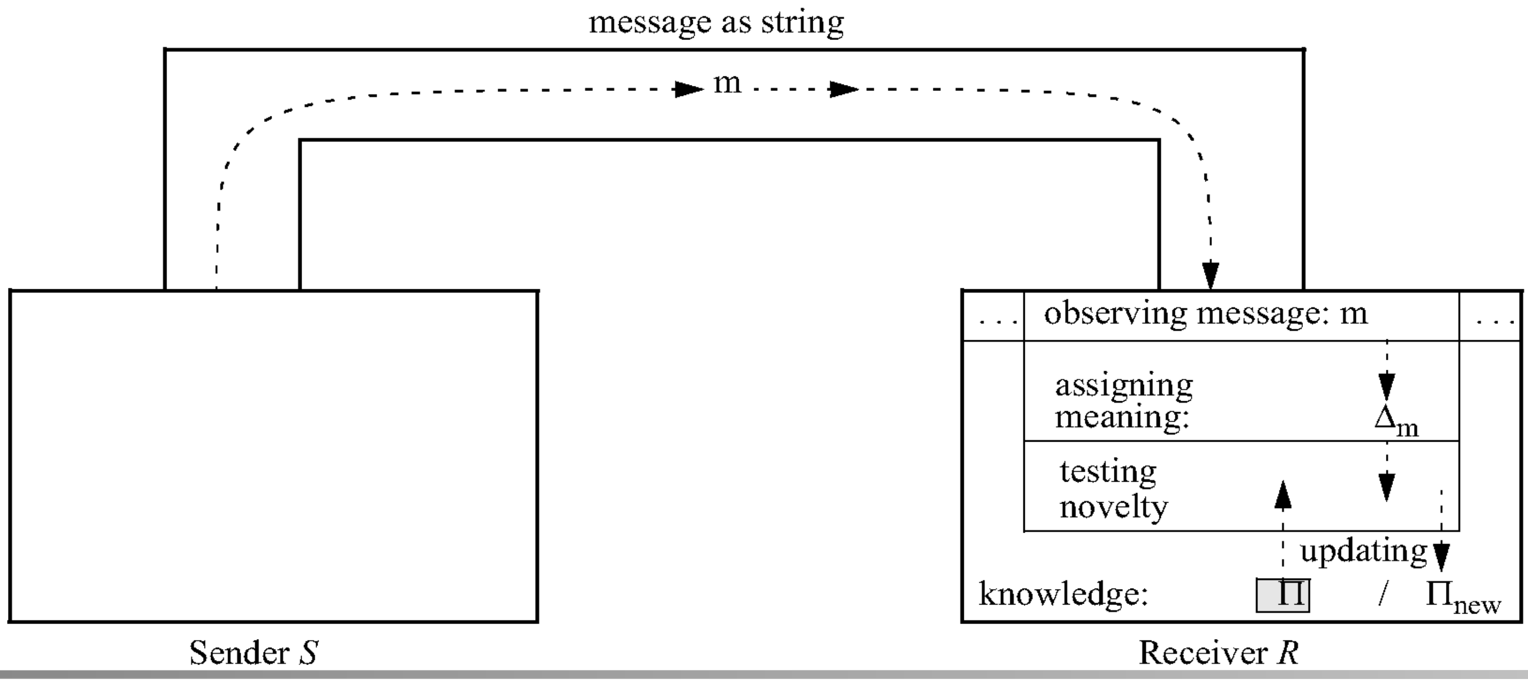
Information flow

- a transmitted message, seen as a string (of letters and, ultimately, of 0's and 1's), is not necessarily *meaningful* concerning content for a receiver or any other *observer*
- it may happen and can even be sensible that an observed string appears random and without information:
  from the point of view of the observer, the message transmission has *not* caused an information flow
- in other cases, an observer succeeds in assigning a meaning to the observed string, roughly in the following sense:
  he determines an assertion expressing the truth of some aspect of his considerations;
  if, additionally, the observer has newly learnt this truth, then the message transmission has caused an *information flow* from the observer's point of view

Information flow based on message transmission

1. observing a message: consider a string m
2. assigning meaning: determine a sentence Δ_m
3. expressing knowledge: form presupposition Π as a collection of sentences

   testing novelty: infer whether Π implies Δ_m
4. updating the knowledge: if novel (not implied), add Δ_m to Π and reorganize, resulting in Π_{new}.



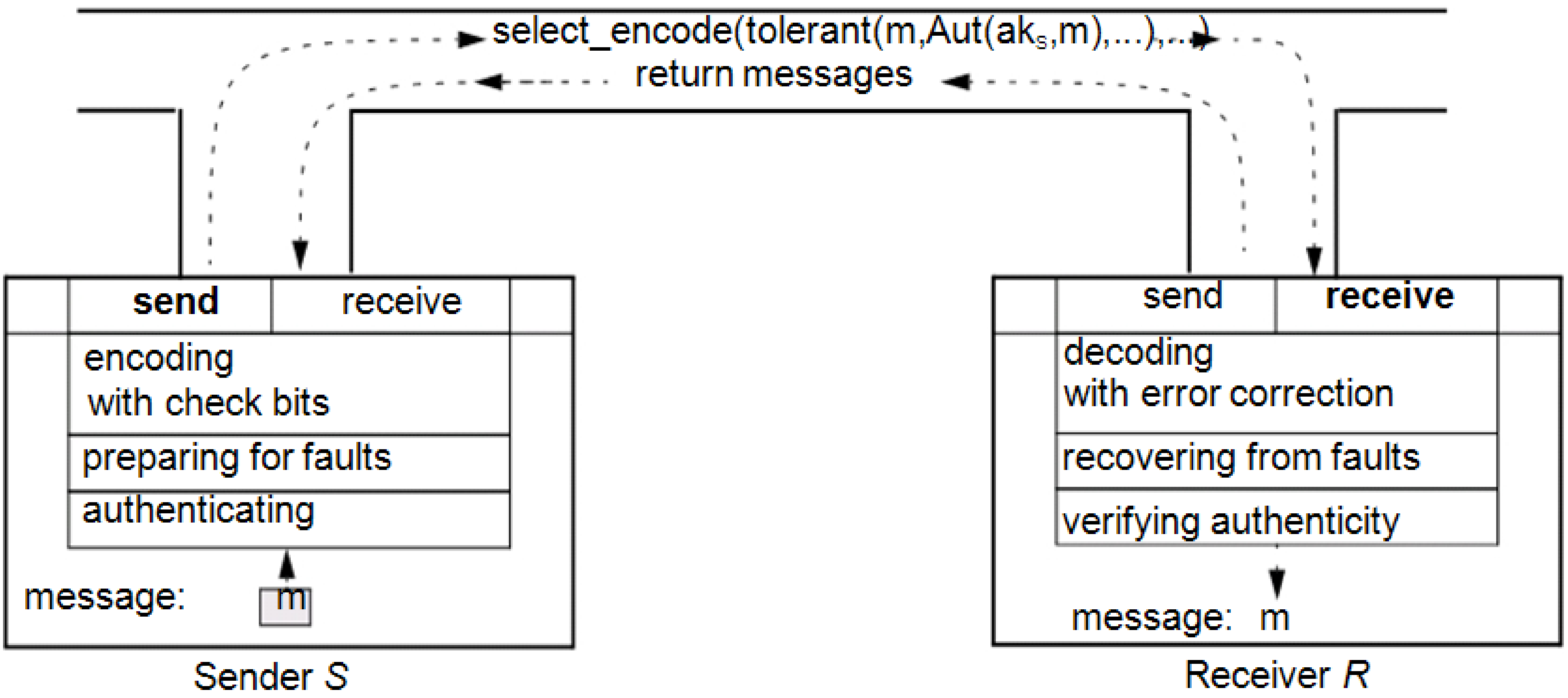
Information flow and message transmission

- a message transmission does not necessarily cause an information flow for any observer
- sometimes an observer has to infer implications in order to let a message transmission appear as an information flow from his point of view
- for such an inference, the observer can exploit a priori knowledge such as a previously acquired key
- for an actual inference,
- the observer needs appropriate computational means

Inspection and exception handling: basic approach

- a message transmission can be accidentally disturbed or deliberately distorted, with the effect that the receiver observes a modified or even forged message
- as a provision against such unfortunate events,
  - senders generate redundancy in the form of *auxiliary objects*, in particular:
  - additional (check) bits for encoding
  - Cryptographic exhibits for authentication
- Participants agree on protocols to exploit the redundancy, in particular:
  - To detect and correct errors for decoding
  - To detect and recover from faults for fault-tolerant computing
  - To defect forgeries for authenticity verification

Inspection and exception handling: summary



Security interests in terms of message transmission / information flow

- each participant should express his *interests*
- *with respect to the service considered*
- *(here: message transmission /information flow)*
- *some interests mainly expect reliable correctness, i.e.,
correct execution of the specified service even in the presence of threats,
and maybe also additional evidence for actual executions*
- *other interests mainly require confinement, i.e.,
that nobody can misuse the service for unwanted effects*

Threats: originators and causes

originators

- the interest holder himself
- participants directly involved in the service
- paricipants woh have implemneted the service
- other participants who are authorized to share the computing system
- intruders from outside
- manufactures, verndors and administrator

originators might threat the service

- harmlessy and accidenty

causes might range from

- improper requirements, through
- faulty implementations or
- wrong administration, to
- unfortunate external events

Trust and threats

- while interacting, one participant might see another one both as a wanted partner and as a potentially *threatening* opponent
- at least some limited *trust* has to be assigned to some participants involved
- components of a computing system might fail, but a user has to trust at least some components

Crucial points of multilateral security

- the trust needed should be minimized
while simultaneously maximizing the achievable functionality, thereby facing the potential threat from the untrusted parts
- each participant should autonomously assign trust at their own discretion
- as far as possible, assigned trust should be justified, and the assigning participant should have the power to verify the trustworthiness and to control the actual behavior of the trusted realm

Confident and optimistic approach

the administrator chooses relatively weak security mechanisms, roughly expecting the following:

- at relatively low cost,
- only slightly affecting the standard operations,
- most of the anticipated threats are effectively covered,
- but exceptional violations (hopefully rare) might still be possible;
- such violations will, hopefully, manageable or acceptable,
- though potentially at high cost

Provisional and pessimistic approach

the administrator selects relatively strong security mechanisms, roughly expecting the following:

- at relatively high cost,
- greatly affecting the standard operations,
- all anticipated threats are effectively covered

Optimistic approach versus pessimistic approach

- cheap *versus* expensive
- basically unaffected standard operations *versus* an essential security overhead
- approximate *versus* complete coverage of threats
- toleration *versus* strict avoidance of exceptional violations

example: access control

optimistic: we audit all activities and,
taking random samples or in cases of suspicion,
analyze the audit trail for violations only afterwards

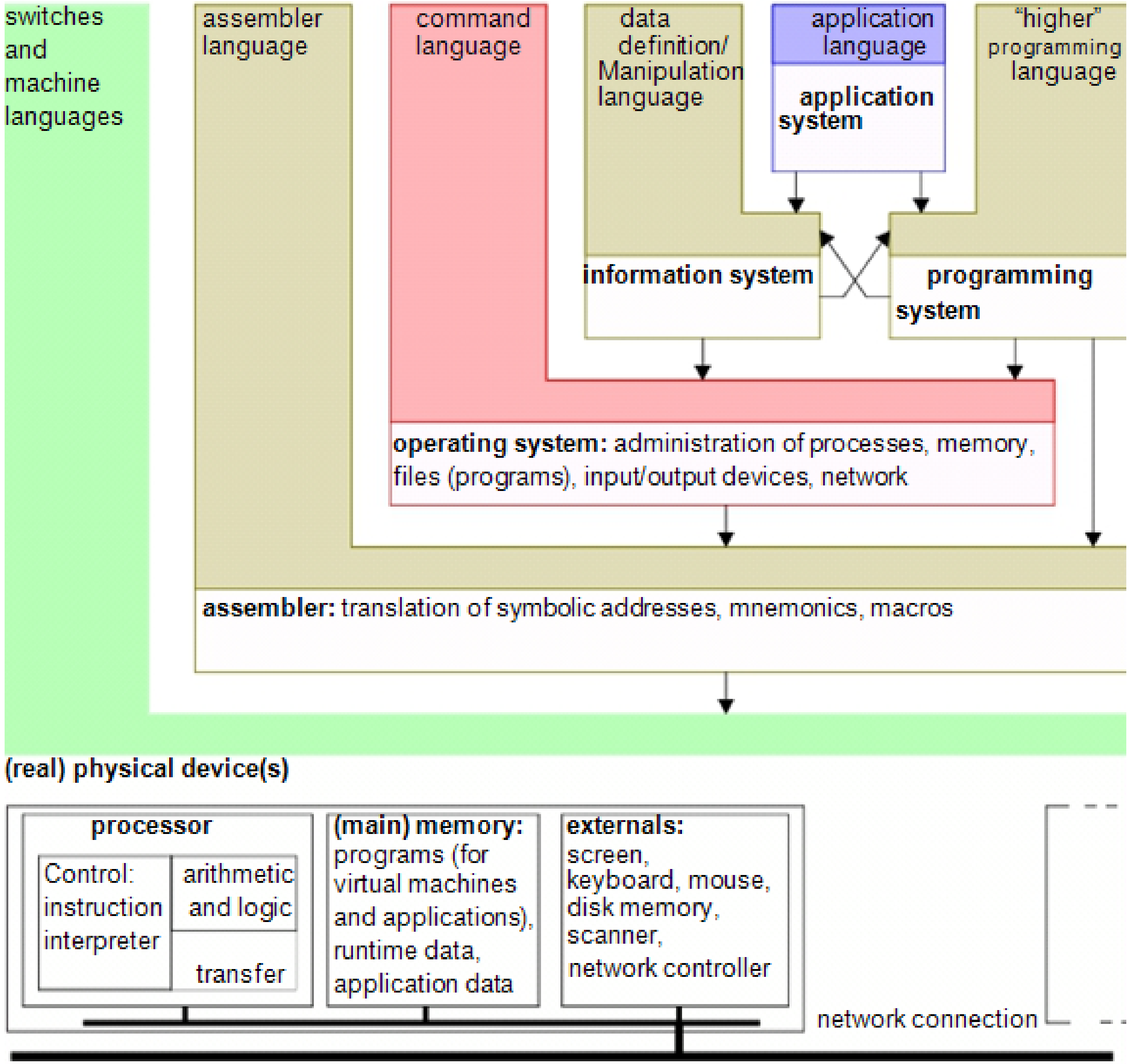
pessimistic: we fully control all requests for activities and
decide them in advance

example: trading

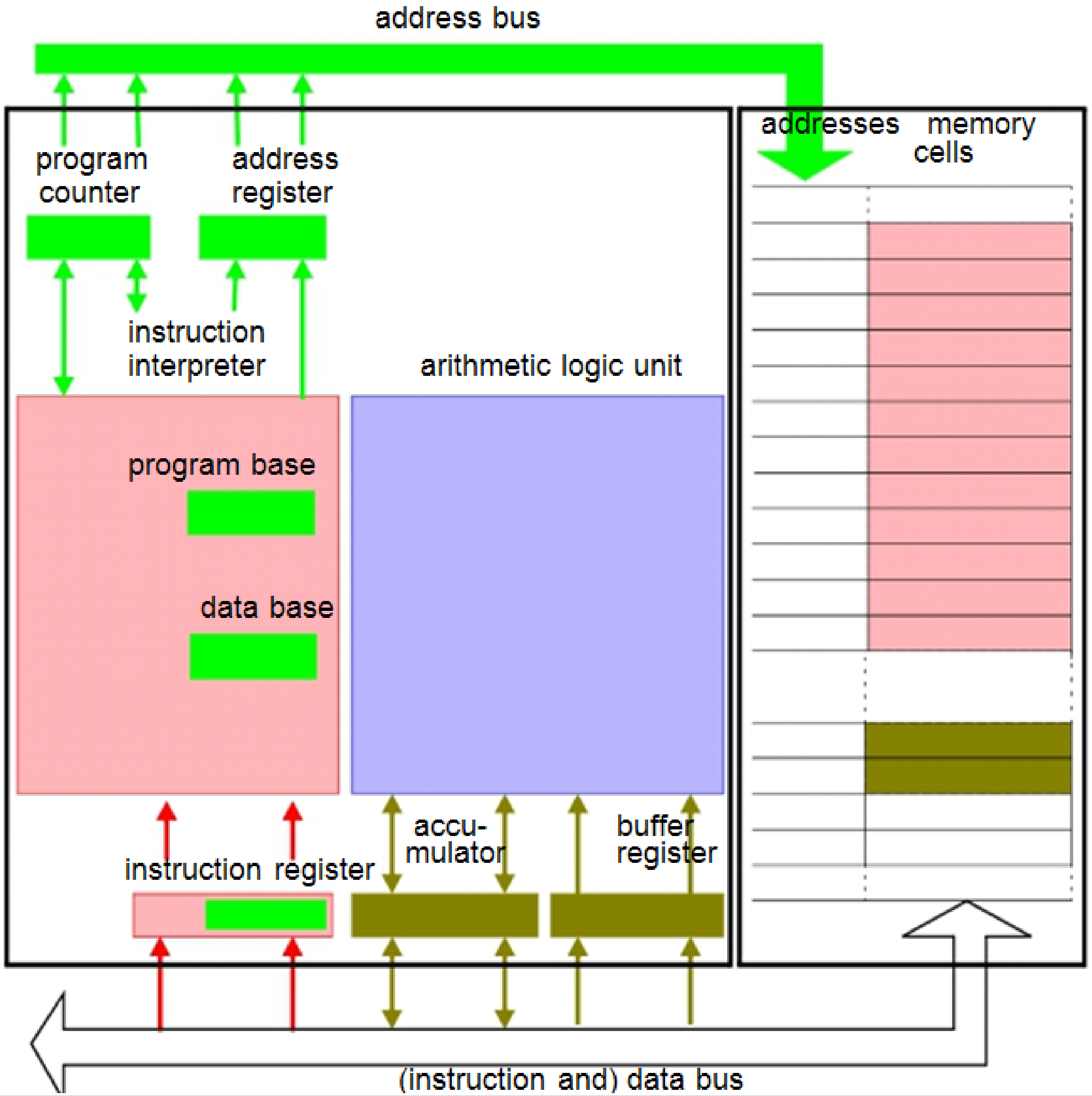
optimistic: cooperating participants issue exhibits by themselves,
which are subject to later evaluation by a trusted third party
only in the case of disputes

pessimistic: every trade is mediated and supervised by a trusted notary

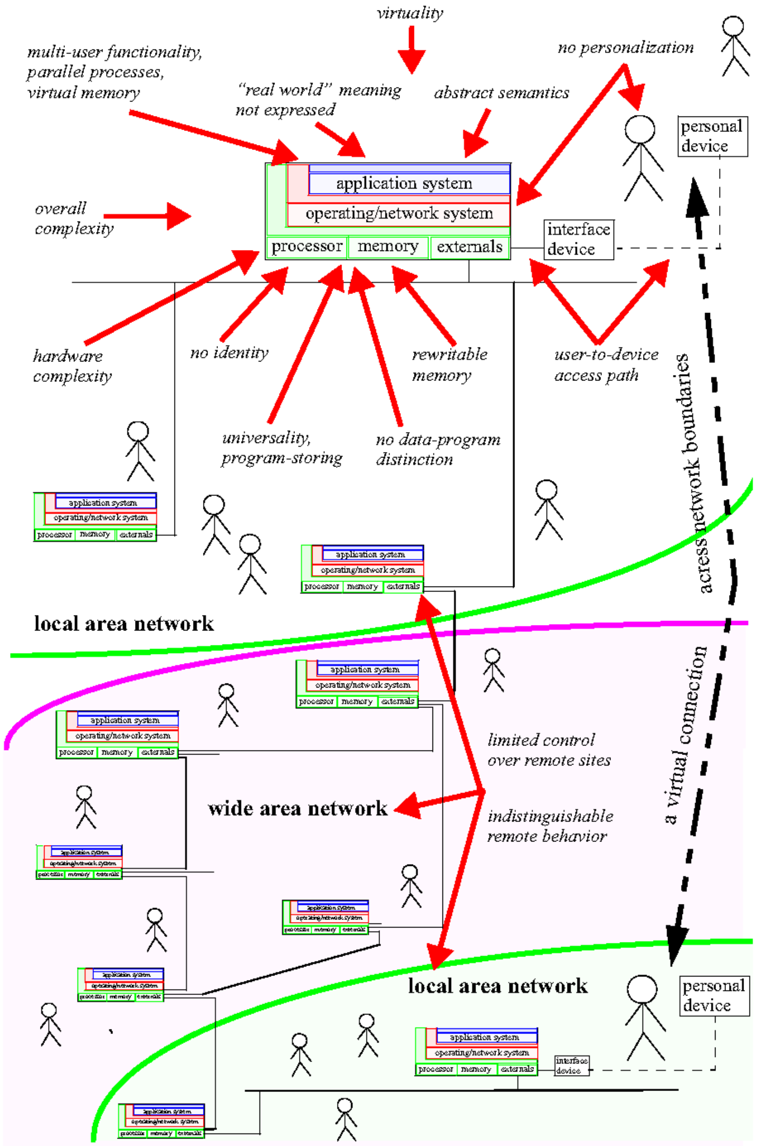
Computing system: layered design



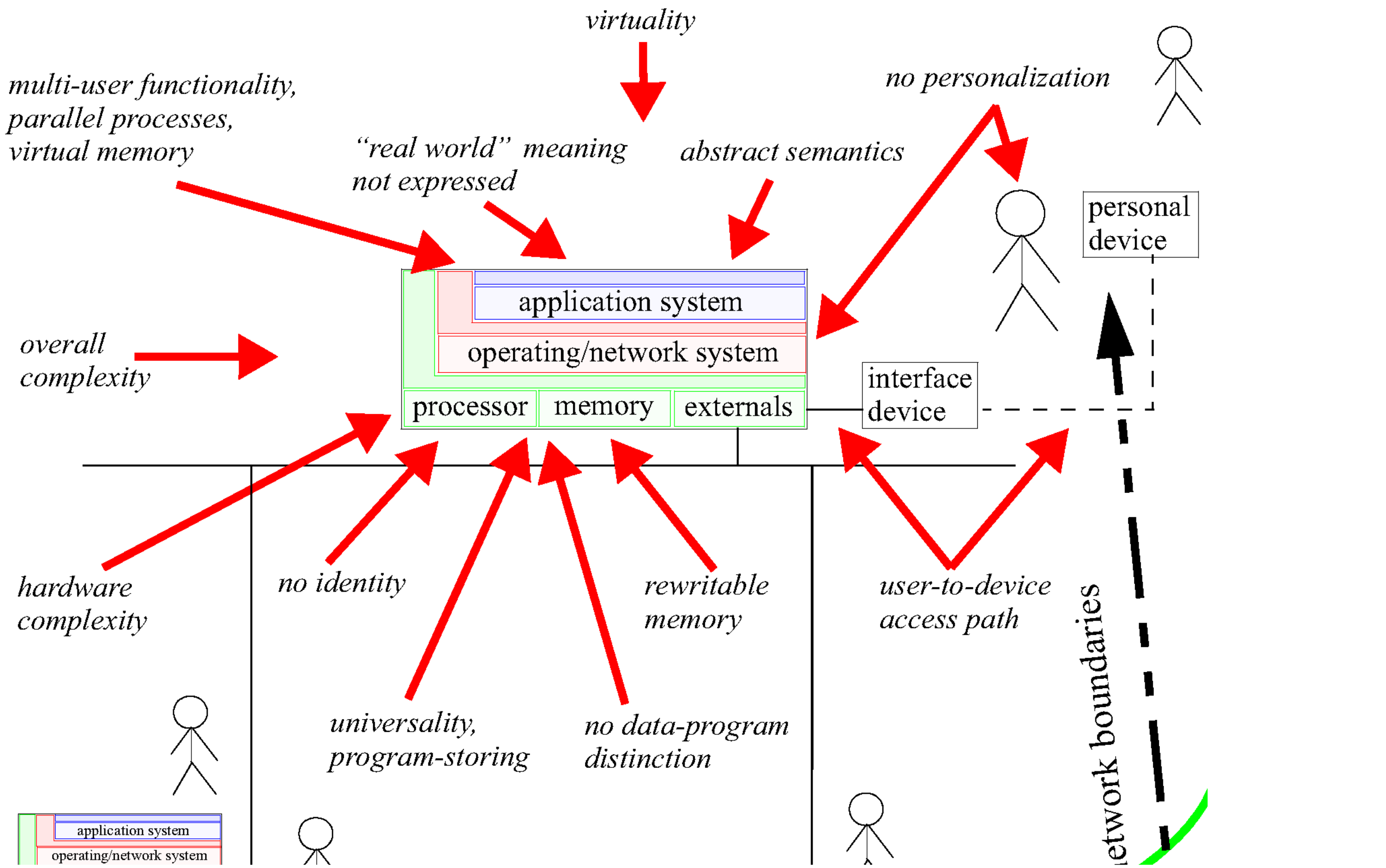
Internal structure of a processor and its memory



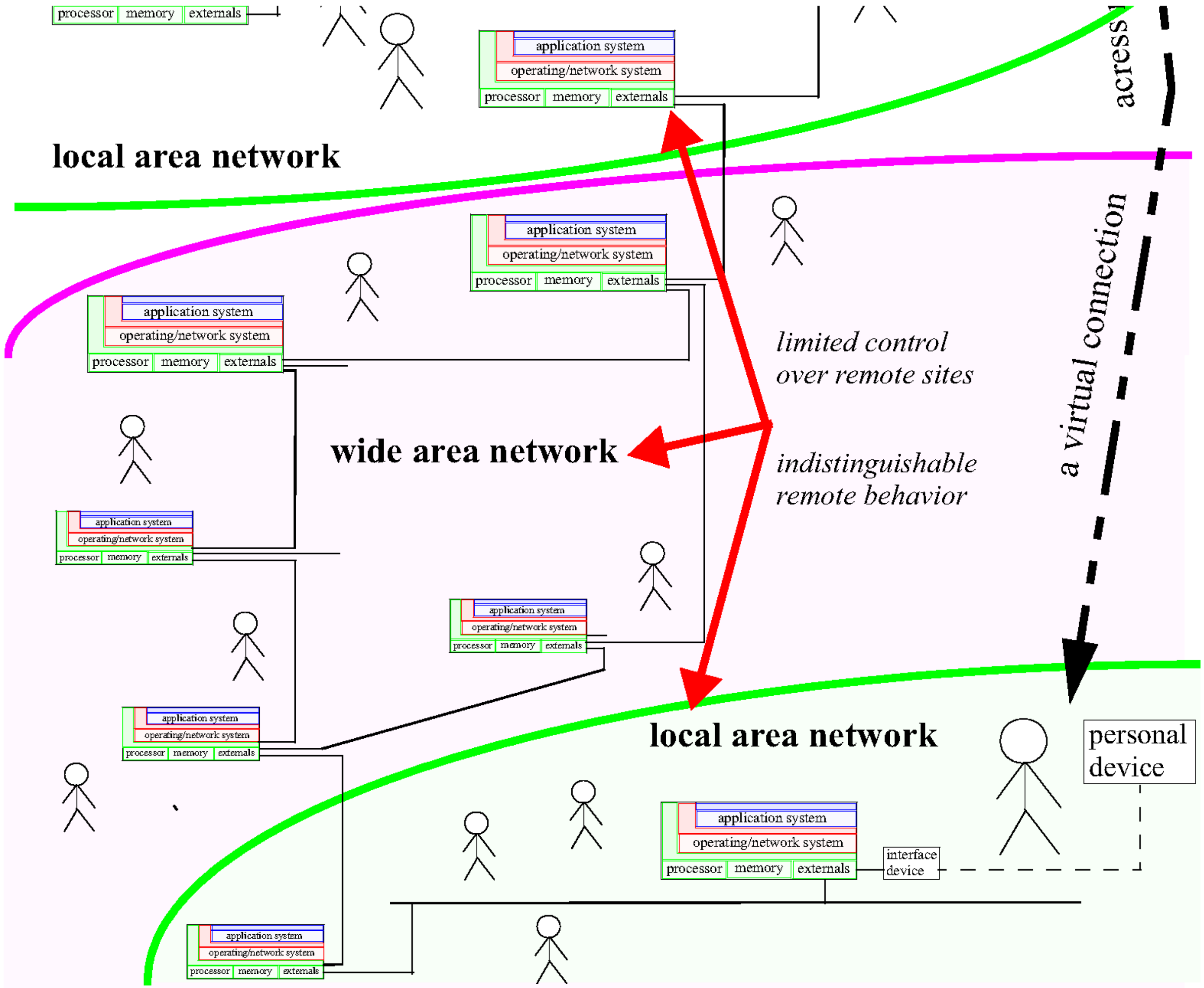
Features of computing and basic vulnerabilities: overview



Features of computing and basic vulnerabilities: one component



Features of computing and basic vulnerabilities: networks



virtuality	"virtual security" corrupted or circumvented in supporting layers
overall complexity	no global, complete understanding; unexpected interferences
universality, program-storing	imposed (malicious) "computable will"
processors without identity	masquerades
devices without personalization	masquerades, repudiated human-device binding
no data-program distinction	program (self-)modification (buffer overflow attacks)
rewritable memory	program and data modification
hardware complexity	hidden functionality, covert channels
user-to-device access path	exposed attack target
multi-user functionality, parallel processes and virtual memory	unintended interferences by resource sharing
abstract semantics of virtual layers	incorrect translation, non-captured but security-relevant aspects
"real-world" meaning not expressed	unperceived attack possibilities
seemingly restricted functionality	universality by simulation
(identifiable) virtual digital objects represented by bit string	(double spending of coins)
limited control over remote sites	remote activities only derivable by inferences
indistinguishable remote behaviour	eavesdropping, message manipulation and forgery, (malicious) message production

Summary

- Security requirements:
  - Confidentiality
  - Integrity
  - Secrecy
  - Authenticity
  - Non-repudiation
  - Secure Information Flow
- Fundamental aspects of security
- Protection rules for personal data
- Various construction principles