Q1: Is it possible to build arbitrarily reliable Systems out of unreliable components?

Q2: Can we achieve consensus in the presence of faults (consensus: all non-faulty components agree on action)?

Q3: Is there an algorithm to determine for a system with a given setting of access control permissions, whether or not a Subject A can obtain a right on Object B?

2 Models per Question!
Q1: Can we build arbitrarily reliable Systems out of unreliable components?

- How to build reliable systems from less reliable components
- Fault/Error, Failure, Fault, ....) terminology in this lecture synonymously used for “something goes wrong” (more precise definitions and types of faults in SE)
Reliability:

- R(t): probability for a system to survive time t

Availability:

- A: fraction of time a system works
INGREDIENTS OF FT

- Fault detection and confinement
- Recovery
- Repair
- Redundancy
  - Information
  - time
  - structural
  - functional
John v. Neumann
Voter: single point of failure

Can we do better
→ distributed solutions?
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Serial-Systems

Each component must work for the whole system to work.
Parallel-Systems

One component must work for the whole system to work.
Each component must fail for the whole system to fail.

\[ R_{\text{whole}} = 1 - \prod_{i=1}^{m} (1 - R_i) \]
Q1/MODEL1: ABSTRACT MODEL

Serial-Parallel-Systems

\[ R_{\text{whole}} = 1 - \prod_{j=1}^{m} \left( 1 - \prod_{i=1}^{n} R_{i,j} \right) \]
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Fault Model

„Computer-Bus-Connector“ can fail such that Computer and/or Bus also fail

=> conceptual separation of components into

Computer, Bus: can fail per se

CC: Computer-Connector fault also breaks the Computer

BC: Bus-Connector fault also breaks Bus
Q1/MODEL1: CONCRETE MODEL

1 Buses
1 Computers

Bus 1

Computer 1

CC 1,1

BC 1,1

1 Buses
1 Computers
Q1/MODEL1: CONCRETE MODEL

1 Buses
2 Computers
Q1/MODEL1: CONCRETE MODEL

1 Buses
N Computers

N Computers

1 Buses
Q1/MODEL1: CONCRETE MODEL

M Buses

1 Computers

Computer 1

Bus 1

Bus \( m \)

BC 1,1

BC 1, m

CC 1,1

CC 1, m
Q1/MODEL1: CONCRETE MODEL

M Buses
N Computers

Computer 1
CC 1,1
CC 1,2
CC 1,m

Computer 2
CC 2,1
CC 2,2
CC 2,m

Computer n
CC n,1
CC n,2
CC n,m

Bus 1
BC 1,1
BC 2,1
BC n,1

Bus m
BC 1,m
BC 2,m
BC n,m
Q1/MODEL1: CONCRETE MODEL FOR N,M

\[ R_{\text{whole}}(n, m) = \left(1 - \left(1 - R_{Bus} \cdot R_{BC}^n\right)^m\right) \cdot \left(1 - \left(1 - R_{Computer} \cdot R_{CC}^m\right)^n\right) \]

then: \( R_{CC}, R_{BC} < 1 \): \( \lim_{n,m \to \infty} R(n, m) = \)
Q1/MODEL2: LIMITS OF RELIABILITY

- System built of Synapses (John von Neumann, 1956)
- Computation and Fault Model:
  - Synapses deliver "0" or "1"
  - Synapses deliver with $R > 0.5$:
    - with probability $R$ correct result
    - with $(1-R)$ wrong result
- Then we can build systems that deliver correct result for any (arbitrarily high) probability $R$
Q1: Is it possible to build arbitrarily reliable Systems out of unreliable components?

Q2: Can we achieve consensus in the presence of faults (consensus: all non-faulty components agree on action)?

Q3: Is there an algorithm to determine for a system with a given setting of access control permissions, whether or not a Subject A can obtain a right on Object B?

2 Models per Question!
Q2: Can we achieve consensus in the presence of faults all non-faulty components agree on action?

- all correctly working units agree on result/action
- agreement non trivial (based on exchange of messages)
Q2/MODEL 1: “2 ARMY PROBLEM”

- p,q processes
  - communicate using messages
  - messages can get lost
  - no upper time for message delivery known
  - do not crash, do not cheat
- p,q to agree on action (e.g. attack, retreat, ...)
- how many messages needed?
- first mentioned: Jim Gray 1978
Result: there is no protocol with finite messages

Prove by contradiction:

- assume there are finite protocols (mp --> q, mq --> p)*
- choose the shortest protocol MP,
- last message MX: mp --> q or mq --> p
- MX can get lost
- => must not be relied upon => can be omitted
- => MP not the shortest protocol.
- => no finite protocol
Q2/MODEL 2: “BYZANTINE AGREEMENT”

n processes, f traitors, n-f loyals

- communicate by reliable and timely messages (synchronous messages)
- traitors lye, also cheat on forwarding messages
- try to confuse loyals
Goal:

- loyals try to agree on non-trivial action (attack, retreat)
- non-trivial more specific:
  - one process is commander
  - if commander is loyal and gives an order, loyals follow the order otherwise loyals agree on arbitrary action
Q2/MODEL 2: “BYZANTINE AGREEMENT”

3 Processes: 1 traitor, 2 loyals

- Commander
  - attack Lieutenant
  - attack Lieutenant
- Lieutenant
  - he said: retreat
- Lieutenant
3 Processes: 1 traitor, 2 loyals

=> 3 processes not sufficient to tolerate 1 traitor
Q2/MODEL 2: “BYZANTINE AGREEMENT”

4 Processes

- Commander
- Lieutenant 1
- Lieutenant
- Lieutenant

Actions:
- Attack
- Retreat

He said:
- Attack
- Retreat
Q2/MODEL 2: “BYZANTINE AGREEMENT”

4 Processes

all lieutenant receive x, y, z => can decide

**General result:** 3 f + 1 processes needed to tolerate f traitors
Q1: Is it possible to build arbitrarily reliable Systems out of unreliable components?

Q2: Can we achieve consensus in the presence of faults (consensus: all non-faulty components agree on action)?

Q3: Is there an algorithm to determine for a system with a given setting of access control permissions, whether or not a Subject A can obtain a right on Object B?

2 Models per Question!
Q3: Is there an algorithm to determine for a system with a given setting of access control permissions, whether or not a Subject A can obtain a right on Object B?

- given a system of entities, acting as subjects and objects
- subjects performs operations on objects
- dynamic: subjects and objects are created and deleted
- access control permissions between entities can be changed according to some rules
higher level models:
- Bell La Padula,
- Chinese wall

access control:
1) ACM-based operations
2) take grant
**MECHANISMS: ACCESS CONTROL MATRIX**

<table>
<thead>
<tr>
<th>Subjects</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td>O</td>
</tr>
<tr>
<td>Entities</td>
<td>E = S ∪ O</td>
</tr>
<tr>
<td>Rights</td>
<td>{read, write, own,...}</td>
</tr>
<tr>
<td>Matrix</td>
<td>S x E x R</td>
</tr>
</tbody>
</table>

**Simple ACM Operations:**
- enter / delete R into cell (s,o)
- create subject / object
- destroy subject / object

**Matrix:**

<table>
<thead>
<tr>
<th></th>
<th>O1</th>
<th>O2</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>r,w,own</td>
<td>r,w</td>
<td>r,w,own</td>
<td>―</td>
<td>r,w</td>
</tr>
<tr>
<td><strong>S2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>r,w</td>
<td>r,w,own</td>
<td>―</td>
<td>r,w,own</td>
<td>r</td>
</tr>
<tr>
<td><strong>S3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>r,w</td>
<td>r</td>
<td>w</td>
<td>―</td>
<td>r,w,own</td>
</tr>
</tbody>
</table>

ref MB: chapter 2.2
ACM

- Access Control List (ACL)
- Capabilities

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>O2</td>
<td>S1</td>
</tr>
<tr>
<td>r,w,own</td>
<td>r,w</td>
<td>r,w,own</td>
</tr>
<tr>
<td>r,w</td>
<td>r,w,own</td>
<td>--</td>
</tr>
<tr>
<td>r,w</td>
<td>r</td>
<td>w</td>
</tr>
</tbody>
</table>

ref MB: chapter 2.2
Define Protection Mechanisms of an Operating System in terms of sequences of simple ACM operations. Only such defined mechanism provided by the OS can be used to manipulate ACM.
“Leakage”: an access right is placed into S/O that has not been there before. It does not matter whether or not that is allowed.

Is leakage decidable?
Define OS-Mechanisms by simple ACM-Operations:

example:

UNIX create file (S1,F)
create object
enter own into A(S1,F)
enter read into A(S1,F)
enter write into A(S1,F)

ref MB: chapter 2.2
Example:

UNIX chmod -w (S2,F)
if own ∈ A(caller,F)
then delete w in A(S2,F)

Q3:
Given an OS with a ACM-based description of protection mechanisms is “Leakage” decidable for any R in A(x,y) ?

ref MB: chapter 2.2
Q3/Model 1: Decidability of Leakage

Decidable

- no subjects/objects can be created
- only one primitive ACM operation per OS-Mechanism by exhaustive search!

Q3 in general:

- undecidable (proof: reduction to Turing machine)

ref MB: chapter 3
ACM

- Access Control List (ACL)
- Capabilities

ref MB: chapter 2.2
Q3/MODEL 2: “TAKE GRANT”

Directed Graph:

Subjects: ●
Objects: ○
Either S or O: ✗

x has capability on Y with set of rights $\alpha$ on y:

\[ x \xrightarrow{\alpha} y \]

- take right
- x has cap with set of rights $\tau$ that includes t

\[ x \xrightarrow{t} y \]

- grant right
- x has cap with set of rights $\gamma$ that includes g

\[ x \xrightarrow{g} y \]
Rules:

take rule ($\alpha \subseteq \beta$)

a takes ($\alpha$ to y) from z

grant rule ($\alpha \subseteq \beta$)

ref MB: chapter 3.3
Rules:

create rule

\[ x \text{ create } (\alpha \text{ to new vertex}) y \]

remove rule

\[ x \text{ removes } (\alpha \text{ to}) y \]
CanShare($\alpha, x, y, G_0$):

there exists a sequence of $G_0 \ldots G_n$ with $G_0 \vdash^* G_n$

and there is an edge in $G_n$:

\[
\begin{array}{c}
\bullet \\
\alpha \\
\bullet
\end{array} \quad \xrightarrow{\alpha} \quad \begin{array}{c}
x \\
y
\end{array}
\]
take rule ($\alpha \subseteq \beta$)

a takes ($\alpha$ to $y$) from $z$

grant rule ($\alpha \subseteq \beta$)

$z$ grants ($\alpha$ to $y$) to

Question:

ref MB: chapter 3.3
create rule

$z$ takes $(g \text{ to } v)$ from $x$

$z$ grants $(\alpha \text{ to } y)$ to $v$

ref MB: chapter 3.3
CanShare($\alpha, x, y, G_0$):

there exists a sequence of $G_0 \ldots G_n$ with $G_0 \vdash^* G_n$

and there is an edge:

CanShare decidable in linear time!
three questions, 2 models per question, different answers !!!
modeling is powerful
need to look extremely carefully into understanding models !!!
Q1/M1:

Q1/M2:
John v. Neuman, PROBABILISTIC LOGICS AND THE SYNTHESIS OF RELIABLE. ORGANISMS FROM UNRELIABLE COMPONENTS.

Q2: most textbooks on distributed systems