

# Directions in Fault-tolerant and Secure Distributed Algorithms

Felix Gärtner



TU Darmstadt, Germany

[fcg@acm.org](mailto:fcg@acm.org)



# Critical Infrastructures



Quelle: <http://www.cs.virginia.edu/~survive/>

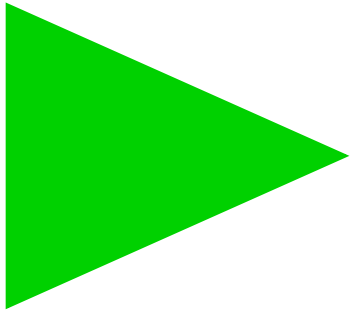


# Complex Computer Systems

Today's computer systems frequently consist of many interacting processes. Here complexity arises due to concurrency, real-time behavior, and heterogeneity.

MPII Research Programme

- . . . and due to **hardware/software faults** and **malicious attackers**.
- Mathematical **modeling** and **analysis** is vital.
- Need to find **suitable abstractions**.



# Distributed Algorithms

- Geographically separated, concurrent processes cooperate to reach a common goal.
- Problems:
  - Lack of common time frame.
  - Lack of common view.
  - Inherent non-determinism.
- Examples: Network protocols (TCP/IP), routing, spanning-tree construction, . . .



# Directions is Distributed Algorithms

(Correct) Distributed Algorithms■



Fault-tolerant Distributed Algorithms■



Secure Fault-tolerant Distributed Algorithms



# Asynchronous System Model

- **Distributed** network nodes communicate via **message passing**.
- Messages can take **arbitrarily long**.
- Nodes can be **arbitrarily slow**.
- Common system model for the **Internet**.
- No faults!



# Properties of Distributed Algorithms

- **System**: state machine/event system with interface.
- **Specification**: functional properties defined on individual executions of the system.
- **Safety** properties: “always . . .”.
- **Liveness** properties: “eventually . . .”.
- Safety and liveness are **fundamental**.



## Example: Mutual Exclusion

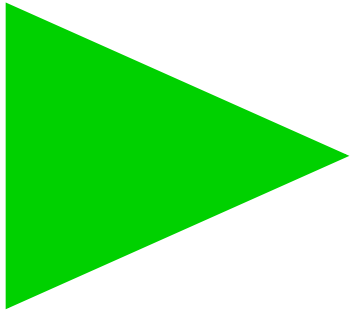
- Research group with **one printer**.
- **Safety**:
  - It is never the case that any two users access the printer at the same time.
- **Liveness**:
  - If a user wants to print something she will eventually succeed.
- How specify **fairness**?





# Correct Distributed Algorithms

- Prove safety using **invariant** arguments.
- Prove liveness using **well-foundedness** arguments.
- A large collection of verified algorithms for standard problems exists:
  - Mutual exclusion, leader election, spanning-tree construction, . . .
- We know many intricacies of system models (e.g., use of randomization in anonymous networks).



## What about Faults?

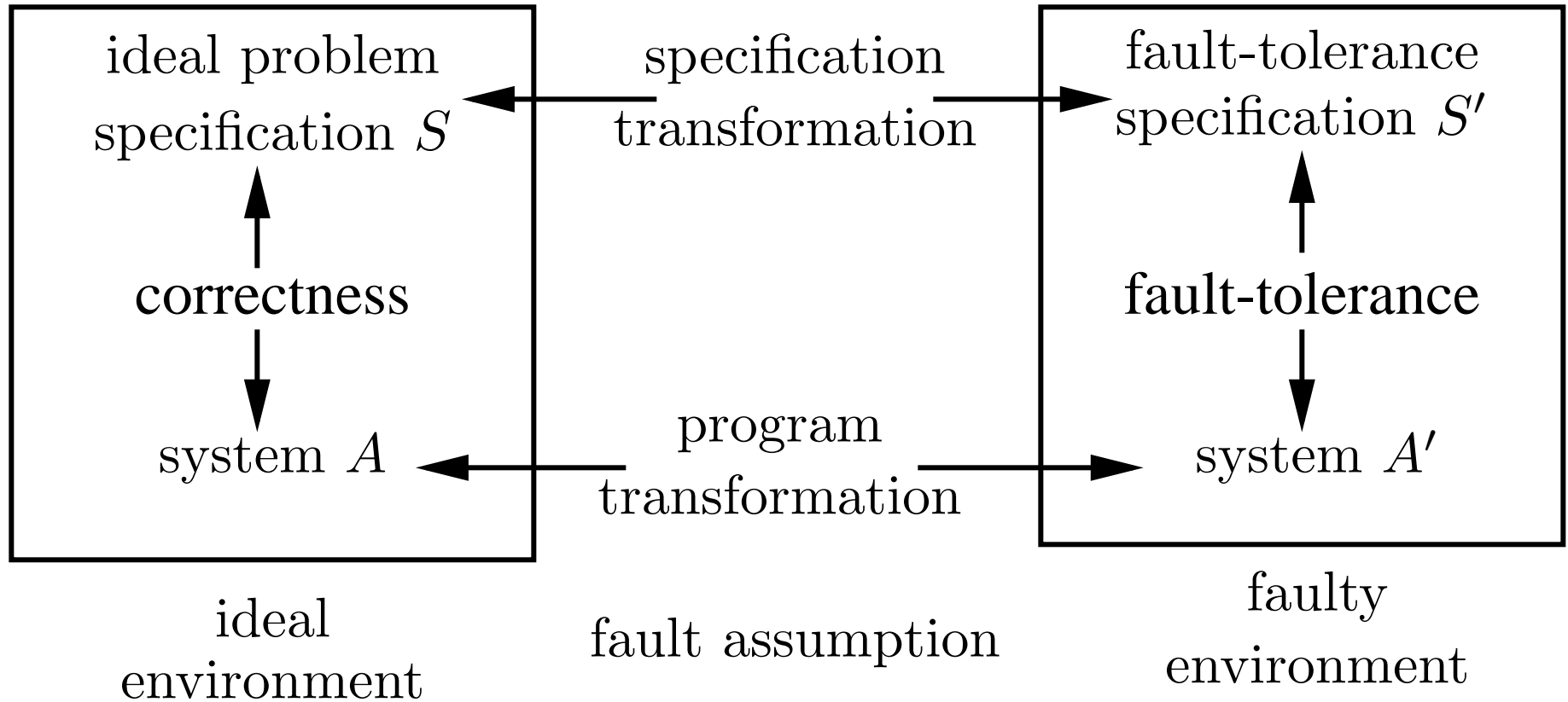
- **Faults** can be:
  - memory perturbation (cosmic rays),
  - link failure (construction works),
  - node crash (power outage), . . .
- Faults can be modeled as **unexpected events**.
- Adding and “removing” state transitions is enough.
- Formalized as a **fault assumption**.

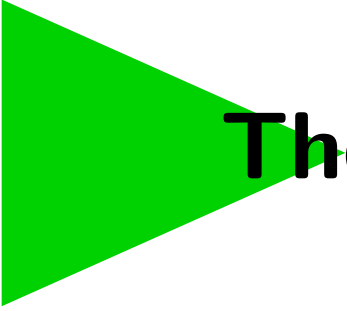


## Fault Tolerance Example

- **Fault assumption**: links and nodes can crash, but network stays connected.
- We want to do **reliable broadcast**:
  - A message which is delivered was previously broadcast (safety).
  - A broadcast message is eventually delivered on all **surviving** machines (liveness).
- Handling gets easier if we relate fault-free environment to faulty environment.

# Fault-tolerant Distributed Algorithms





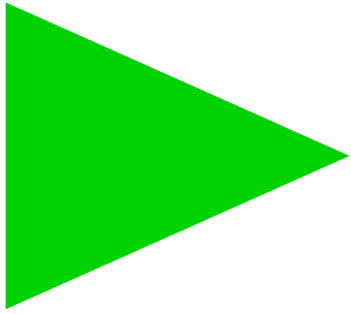
# Theory of Fault-tolerant Systems

- Fault-tolerant program =  
fault-**intolerant** programm + fault-tolerance  
components.
- **Detector**: detects system state.
- **Corrector**: corrects to a system state.
- Abstraction of many known mechanisms.



# Formal Foundations of Fault-tolerance

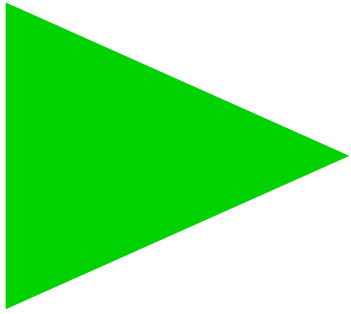
- Theorems:
  - **Detectors** are necessary and sufficient to maintain **safety**.
  - **Correctors** are necessary and sufficient to achieve **liveness**.
- Correctors contain detectors.
- Principle of operation: **redundancy**.



# Theory of Redundancy

- **Redundancy in space** = non-reachable states in the absence of faults.
- **Redundancy in time** = never-executed transitions in the absence of faults.

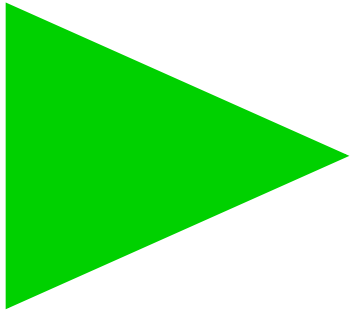
fault-tolerant w.r.t.	necessary
Safety	Redundancy in space
Liveness	Redundancy in time + redundancy in space



# What about Security?

- Many **different aspects** to consider: trust, secrecy, . . .
- Conjecture: Security is **CIA**.
  - **Confidentiality**: non-occurrence of unauthorized disclosure of information.
  - **Integrity**: non-occurrence of inadequate information alterations.
  - **Availability**: readiness for usage.

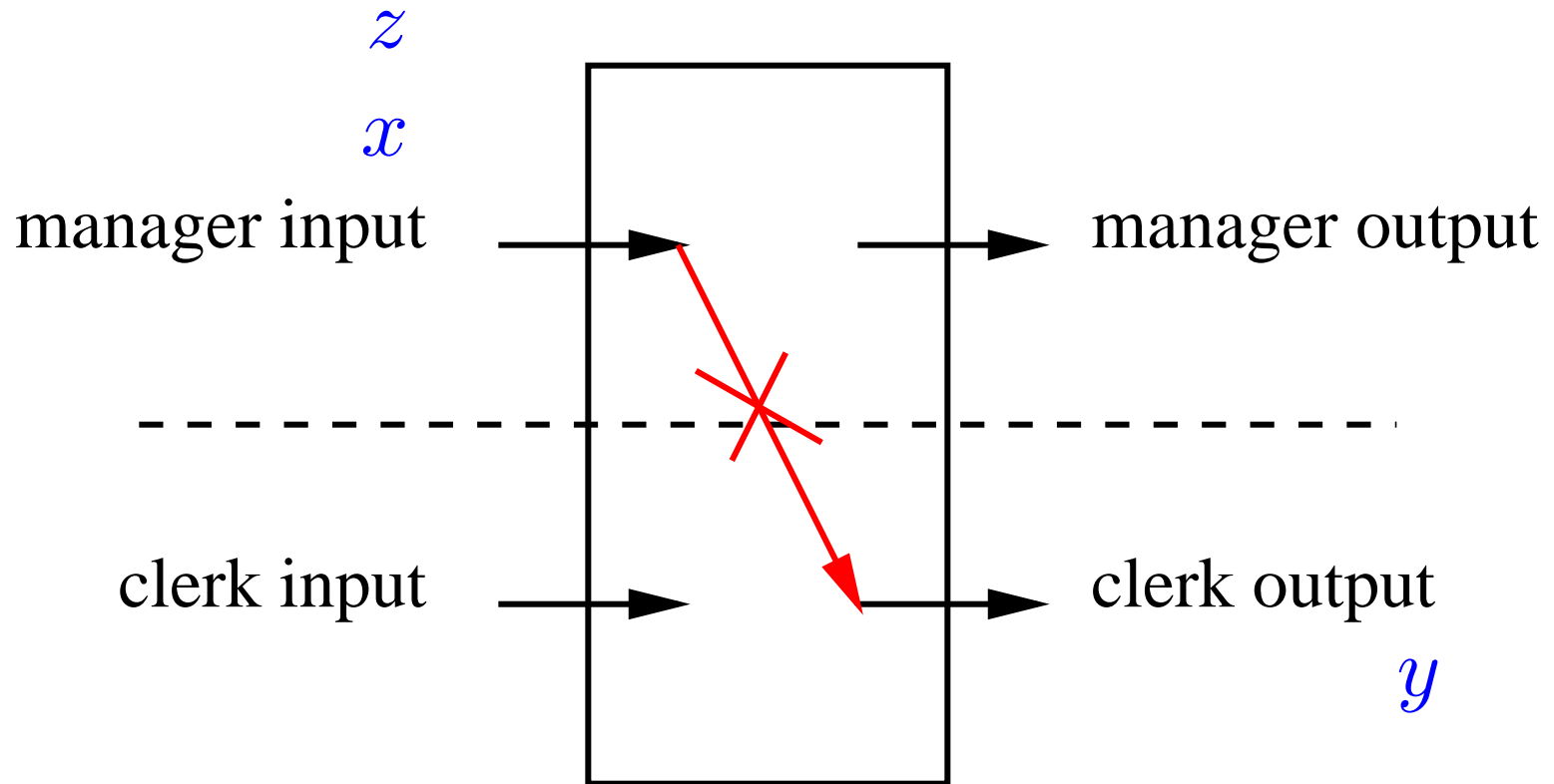




# Security Properties

- We can model a lot of notions from security with safety and liveness:
  - **Access control** is safety.
  - Aspects of **confidentiality** are safety. .
  - Aspects of **integrity** are safety,  
e.g. “no unauthorized change of a variable” .
  - Aspects of **availability** are liveness,  
e.g. “eventual reply to a request” .

# Problems with Information Flow



possible executions:  $x, y$        $z, y$



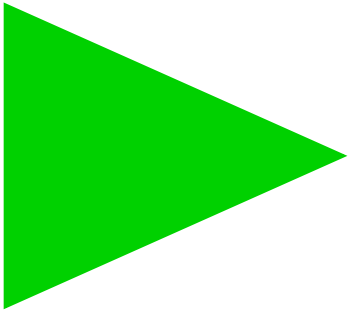


## Higher Level Properties

- Property of the type: if trace  $x, y$  is possible, then trace  $z, y$  must be possible too.
- Usually formalized as **closure conditions on trace sets**:

$$\sigma \in S \Rightarrow f(\sigma) \subseteq S$$

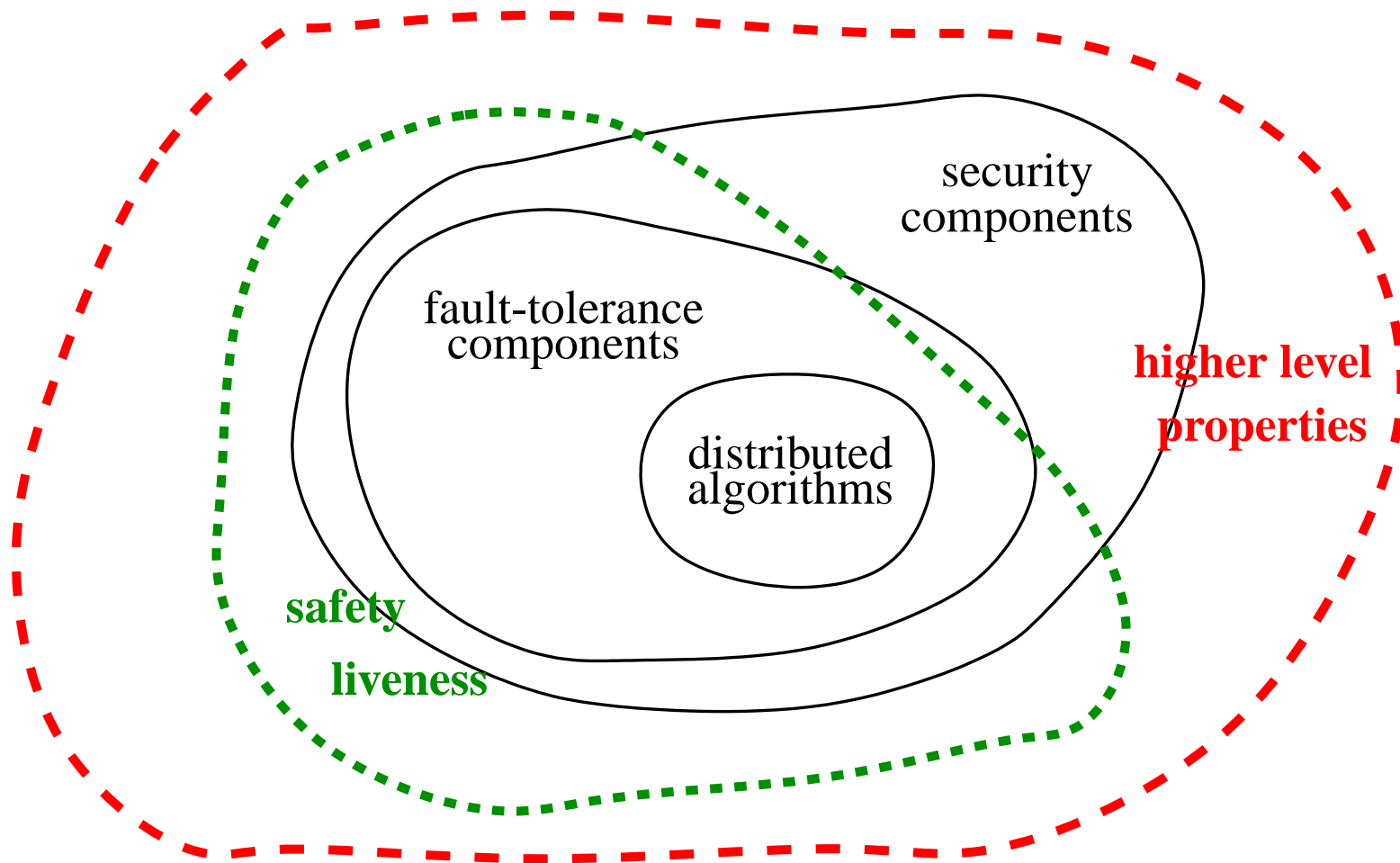
- **Properties of properties**, sets of sets of traces.
- Consequence: Restriction of information flow is **neither safety nor liveness**.

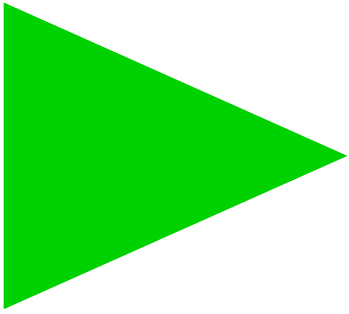


# Open Questions

- Are **higher level properties** enough?
- Relations to **cryptographic definitions of security**?
- Are **attacker assumptions** reasonable (Dolev-Yao)?  
What about “unknown” attacks?
- Promising direction: Develop a **theory of security components** to understand security protocols.

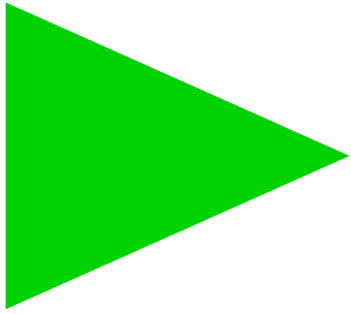
# The World According to F.G.





# Research Agenda

- We need
  - set of **well-understood system models**,
  - set of **reasonable fault/attacker assumptions**,
  - sound **design theories**,
  - algorithmical **building blocks**to understand and build **critical systems**.



# Acknowledgments

- Slides produced using pdfL<sup>A</sup>T<sub>E</sub>X and Klaus Guntermann's PPower4.

## References

- ALPERN, B. AND SCHNEIDER, F. B. 1985. Defining liveness. *Information Processing Letters* 21, 181–185.
- ARORA, A. AND KULKARNI, S. S. 1998. Component based design of multitolerant systems. *IEEE Transactions on Software Engineering* 24, 1 (Jan.), 63–78.
- CACHIN, C., CAMENISCH, J., DACIER, M., DESWARTE, Y., DOBSON, J., HORNE, D., KURSAWE, K., LAPRIE, J.-C., LEBRAUD, J.-C., LONG,

D., MCCUTCHEON, T., MÜLLER, J., PETZOLD, F., PFITZMANN, B., POWELL, D., RANDELL, B., SCHUNTER, M., SHOUP, V., VERÍSSIMO, P., TROUessin, G., STROUD, R. J., WAIDNER, M., AND WELCH, I. S. 2000. Reference model and use cases. Deliverable D1 of the MAFTIA project [MAFTIA ].

CRISTIAN, F. 1985. A rigorous approach to fault-tolerant programming. *IEEE Transactions on Software Engineering* 11, 1 (Jan.), 23–31.

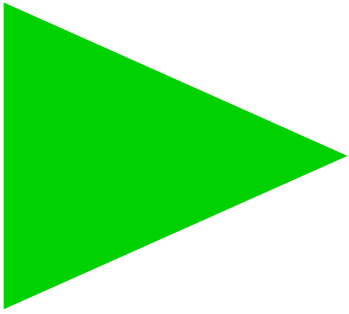
GÄRTNER, F. C. 1998. Specifications for fault tolerance: A comedy of failures. Technical Report TUD-BS-1998-03 (Oct.), Darmstadt University of Technology, Darmstadt, Germany.

GÄRTNER, F. C. 1999. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Computing Surveys* 31, 1 (March), 1–26.

GÄRTNER, F. C. 2001. *Formale Grundlagen der Fehlertoleranz in verteilten Systemen*. Ph. D. thesis, Fachbereich Informatik, TU Darmstadt. forthcoming.



- GRAY, III., J. W. AND MCLEAN, J. 1995. Using temporal logic to specify and verify cryptographic protocols. In *Proceedings of the Eighth Computer Security Foundations Workshop (CSFW '95)* (Washington - Brussels - Tokyo, June 1995), pp. 108–117. IEEE.
- LAPRIE, J.-C. Ed. 1992. *Dependability: Basic concepts and Terminology*, Volume 5 of *Dependable Computing and Fault-Tolerant Systems*. Springer-Verlag.
- MAFTIA. Maftia home – Malicious- and Accidental-Fault Tolerance for Internet Applications. Internet:  
<http://www.newcastle.research.ec.org/maftia/>.
- SCHNEIDER, F. B. 2000. Enforceable security policies. *ACM Transactions on Information and System Security* 3, 1 (Feb.), 30–50.



# Abstract

The failure of a critical computer system can have unpleasant consequences like severe irritation, industrial damage, even loss of human lives. Failures can arise due to hardware and software faults, but also as the result of malicious actions initiated by an attacker of the system. While the former should be dealt with using fault-tolerance mechanisms, the latter should be addressed using approaches from computer security.

In the past, the existing fault tolerance and security problems in practice have often been dealt with using ad hoc methods developed by practitioners in response to urgent development needs. However, many of the questions regarding the underlying principles of fault-tolerant and secure operations have not been sufficiently answered yet. What is needed is a theoretically sound methodological foundation for the design of fault-tolerant and secure systems.

In this talk, I will discuss the main aspects of such a foundation which cover questions of system models, fault and attacker assumptions, design theories of fault-tolerant and secure algorithms, and algorithmical building blocks for paradigmatic problems. I will sketch the state of the art as well as directions for future work in these areas.