Michael Backes

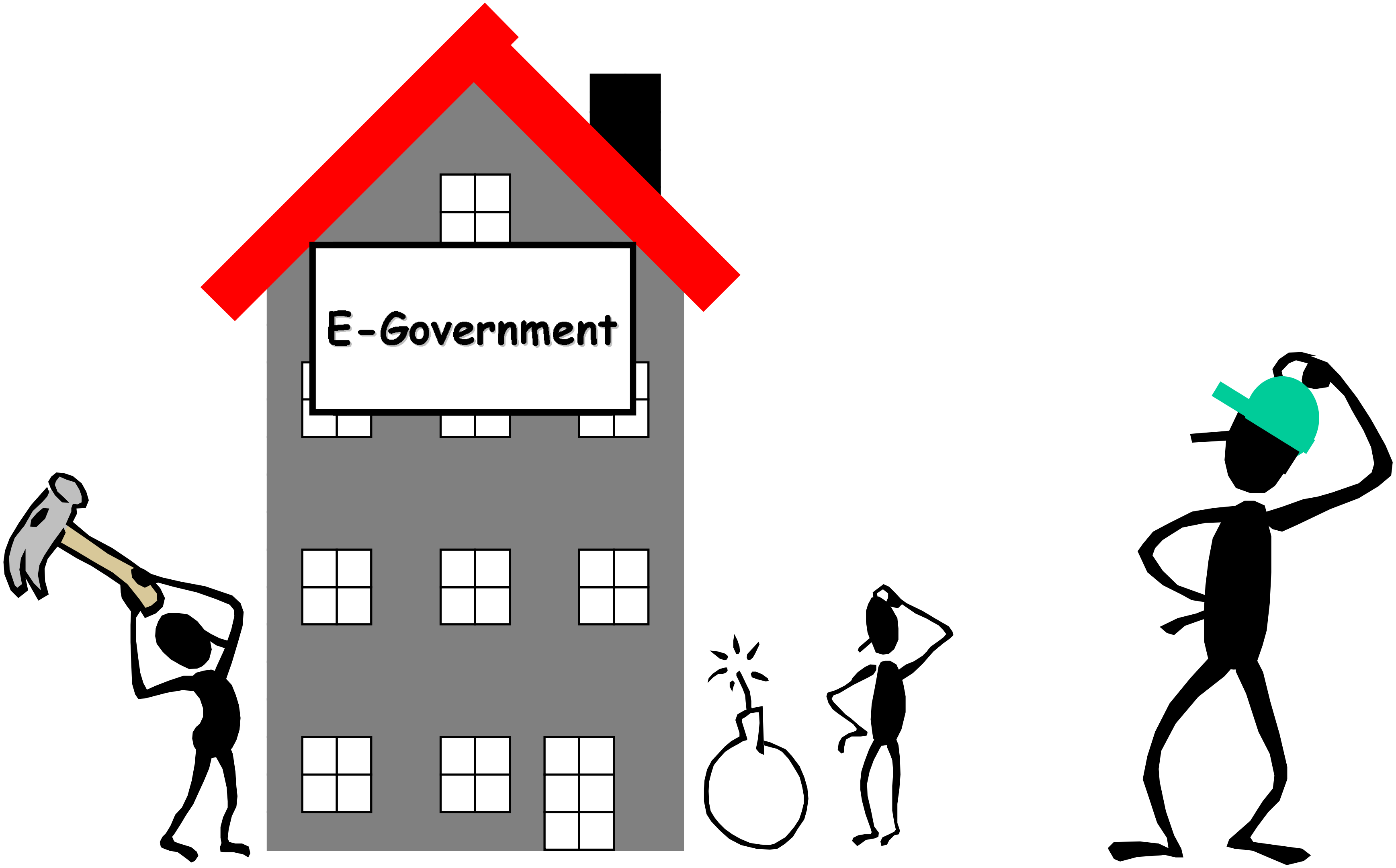
Saarland University, Germany

joint work with Birgit Pfitzmann and Michael Waidner

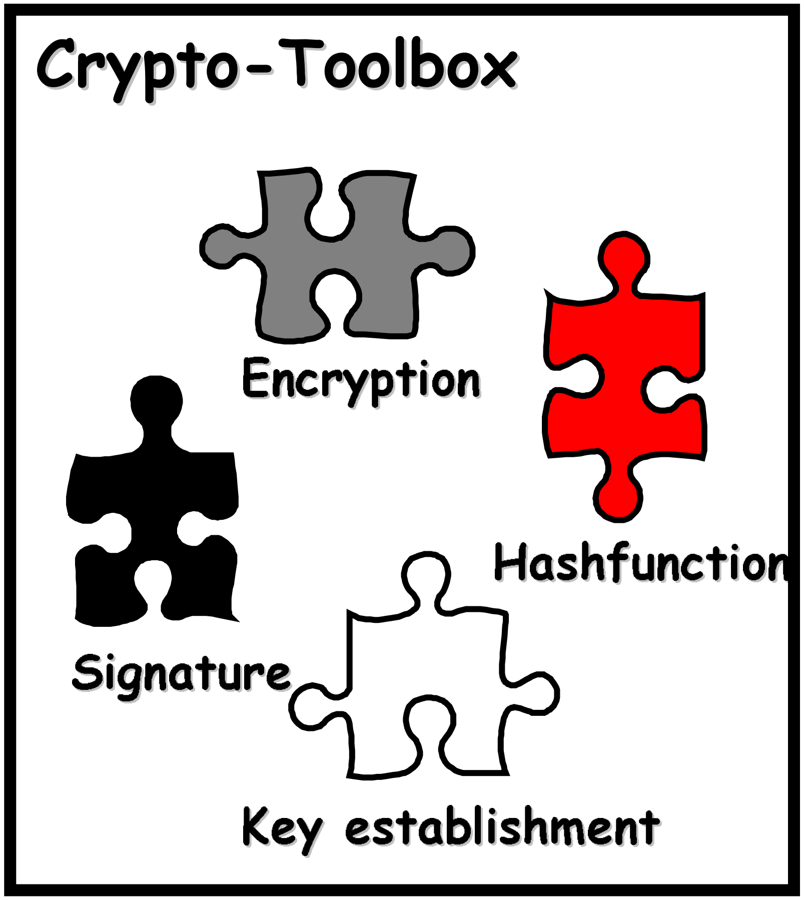
Secure Reactive Systems

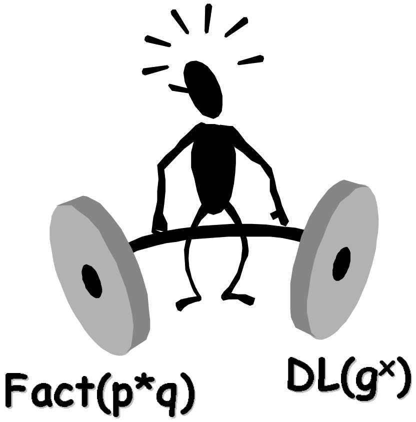
Lecture at Tartu U, 02/27/06

Building Systems on Open Networks

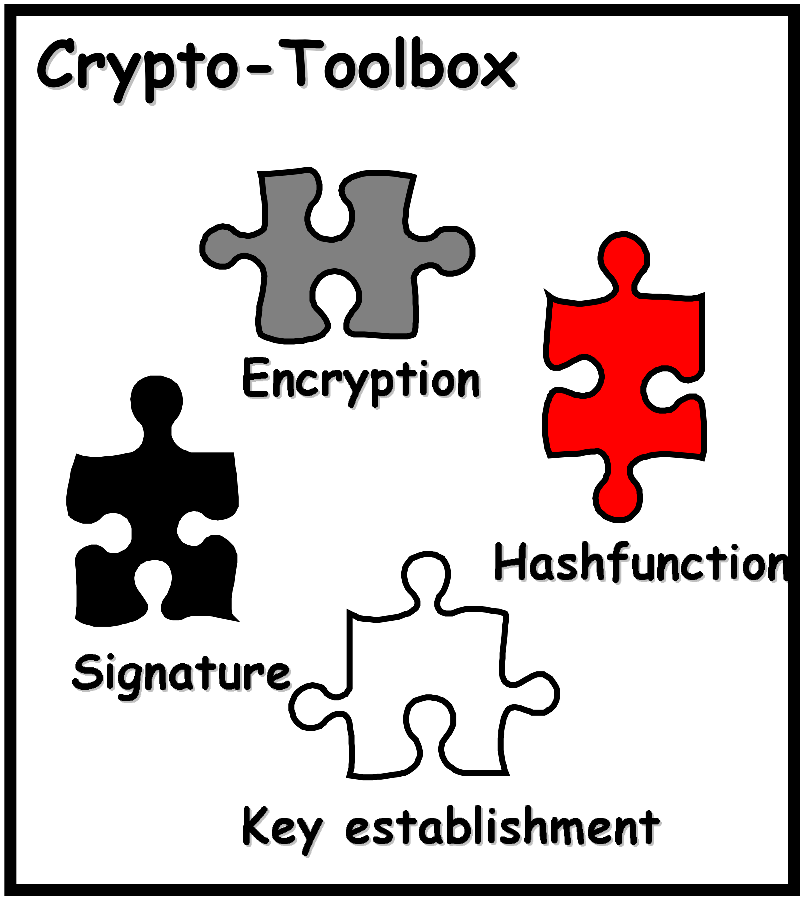


Cryptography: The Details





Cryptography: The Details



Crypto-Toolbox

Encryption

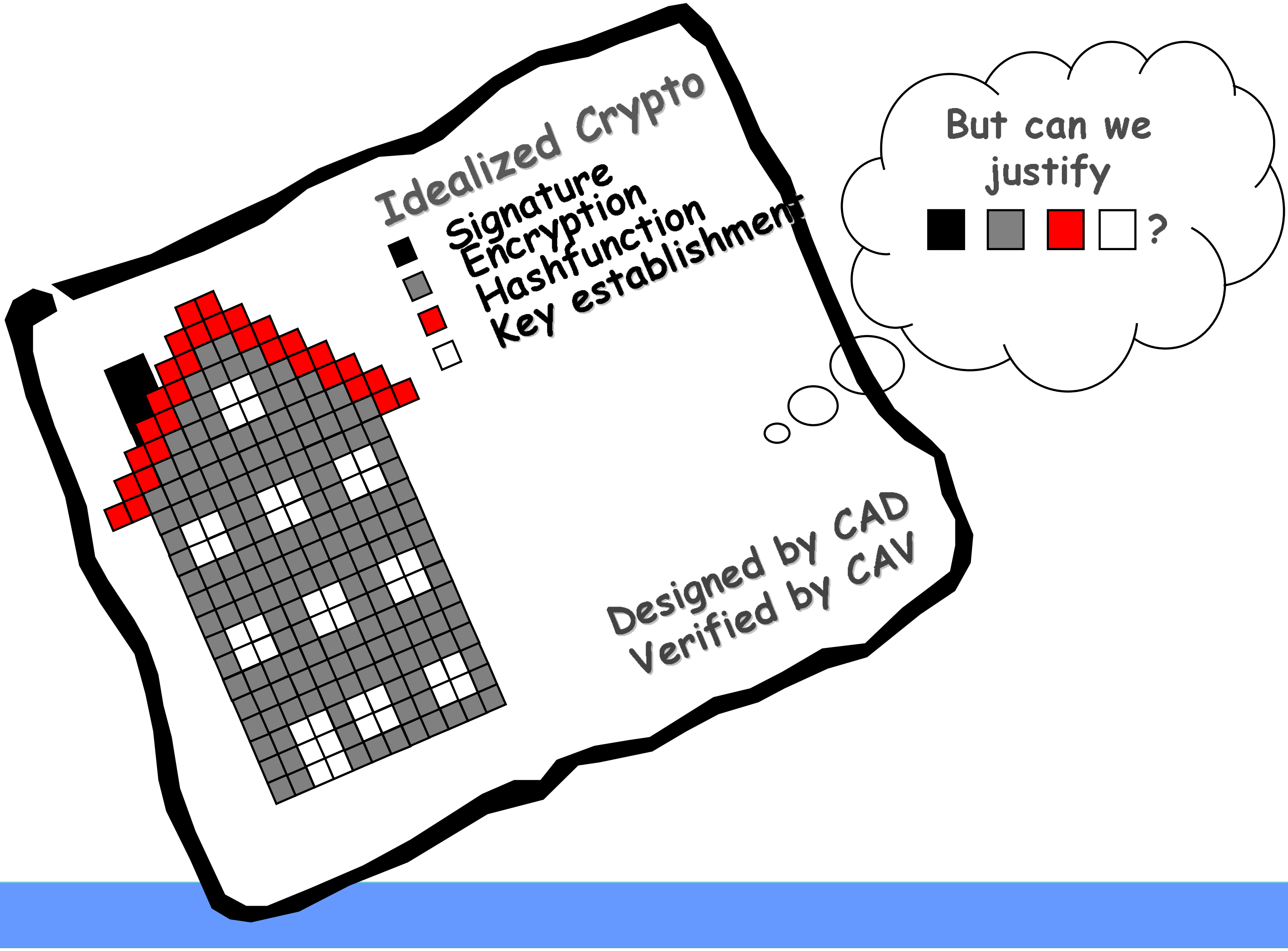
Hashfunction

Signature

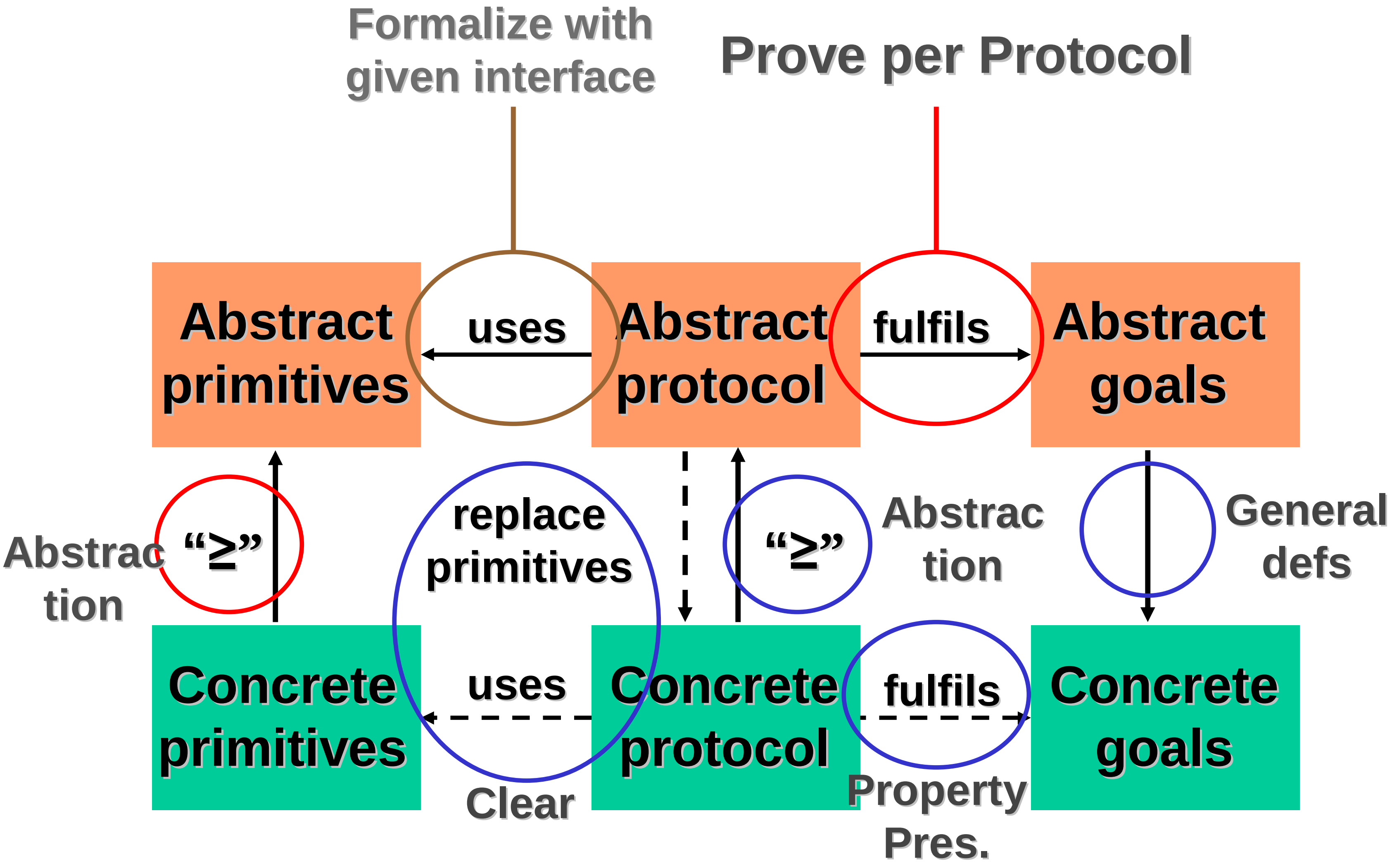
Key establishment

Proof

Formal Methods: The Big Picture



Idea: Sound Abstract Protocol Proofs



Formalize with given interface

Prove per Protocol

Abstract primitives

uses

Abstract protocol

fulfils

Abstract goals

Abstraction

"≥"

replace primitives

"≥"

Abstraction

General defs

Concrete primitives

uses

Concrete protocol

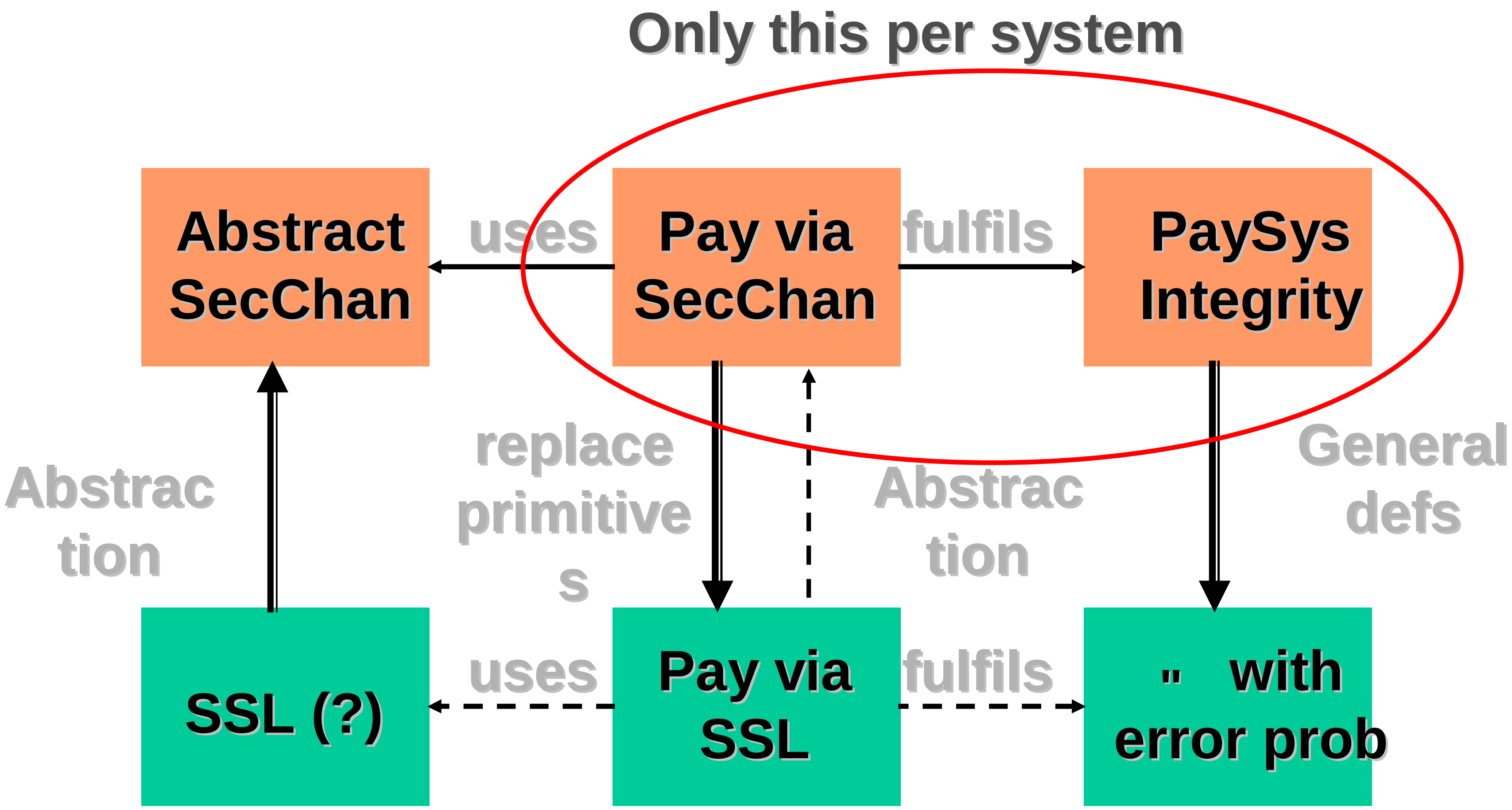
fulfils

Concrete goals

Clear

Property Pres.

Example



Only this per system

Abstract SecChan

uses

Pay via SecChan

fulfils

PaySys Integrity

Abstrac tion

replace primitive s

Abstrac tion

General defs

SSL (?)

uses

Pay via SSL

fulfils

" with error prob

Courses Syllabus

What do we do in this course?

1. Define a rigorous model for reactive systems and give a definition of sound abstraction within this model
2. Show compositionality of the definition (along with some base lemmata) and give concrete examples that satisfy the definition
3. Investigate how specific properties behave under this definition (integrity, confidentiality, liveness, …)
4. Can we even justify symbolic abstractions of crypto with that? Tool support, applications to large protocols, …
5. Limitations of Soundness, and spezialized properties (strong key and message secrecy, etc.)

What do we need for soundly abstracting?

- **Precise system model that permits all “realistic” attacks.**

What do we need for soundly abstracting?

- **Precise system model that permits all “realistic” attacks.**
  - **Network characteristics? synchr./asynchr., reliable, secure, etc.**
  - **Power of the adversary? Passive/active, static/dynamic, secure function evaluation / reactive (!)**
  - **Realistic scheduling**
  - **Which other protocols may run concurrently?**
  - **…**

What do we need for soundly abstracting?

- **Precise system model that permits all “realistic” attacks.**
- **Capable of reasoning about abstractions and realizations at the same time**

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions and realizations at the same time
    - Cryptographic issues: probabilism, error-probabilities, computationsl restrictions, etc.
    - Abstraction issues: Abstract transition functions, distributed-systems aspects, formal calculi, etc.

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions and realizations at the same time

- Mathematically rigorous definition of what a “good” abstraction is

What do we need for soundly abstracting?

- **Precise system model that permits all “realistic” attacks.**
- **Capable of reasoning about abstractions and realizations at the same time**

- **Mathematically rigorous definition of what a “good” abstraction is**
  - **Intuitive**
  - **Should fit to a variety of different abstractions/real protocol classes**
  - **Provable by convenient proof techniques**

What do we need for soundly abstracting?

- **Precise system model that permits all “realistic” attacks.**
- **Capable of reasoning about abstractions and realizations at the same time**

- **Mathematically rigorous definition of what a “good” abstraction is**
- **Should not only hold in isolation but should preserve security under composition.**

What do we need for soundly abstracting?

- **Precise system model that permits all “realistic” attacks.**
- **Capable of reasoning about abstractions and realizations at the same time**

- **Mathematically rigorous definition of what a “good” abstraction is**
- **Should not only hold in isolation but should preserve security under composition.**
  - **(Makes the definition “useful”)**
  - **Make modular analysis of larger protocols possible**

What do we need for soundly abstracting?

- **Precise system model that permits all “realistic” attacks.**
- **Capable of reasoning about abstractions and realizations at the same time**

- **Mathematically rigorous definition of what a “good” abstraction is**
- **Should not only hold in isolation but should preserve security under composition.**
- **Should preserve essentially arbitrary security properties**

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions and realizations at the same time

- Mathematically rigorous definition of what a “good” abstraction is
- Should not only hold in isolation but should preserve security under composition.
- Should preserve essentially arbitrary security properties
  - Integrity, variants of confidentiality, non-interference, poly-time variants of liveness
  - Tight links to properties shown for symbolic abstractions of crypto

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions and realizations at the same time

- Mathematically rigorous definition of what a “good” abstraction is
- Should not only hold in isolation but should preserve security under composition.
- Should preserve essentially arbitrary security properties

- Abstractions should match the intuition for the requirements in mind.

What do we need for soundly abstracting?

- **Precise system model that permits all “realistic” attacks.**
- **Capable of reasoning about abstractions and realizations at the same time**

- **Mathematically rigorous definition of what a “good” abstraction is**
- **Should not only hold in isolation but should preserve security under composition.**
- **Should preserve essentially arbitrary security properties**

- **Abstractions should match the intuition for the requirements in mind.**
  - **Intuitive abstractions, easy to read for non-specialist, thus enabling convenient use in larger protocols**

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions and realizations at the same time

- Mathematically rigorous definition of what a “good” abstraction is
- Should not only hold in isolation but should preserve security under composition.
- Should preserve essentially arbitrary security properties

- Abstractions should match the intuition for the requirements in mind.
- Abstractions should be based on the functionality of the protocol, not on its structure.

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions and realizations at the same time

- Mathematically rigorous definition of what a “good” abstraction is
- Should not only hold in isolation but should preserve security under composition.
- Should preserve essentially arbitrary security properties

- Abstractions should match the intuition for the requirements in mind.
- Abstractions should be based on the functionality of the protocol, not on its structure.
  - Functionalities for large protocol classes
  - Only guarantees matter for larger protocols, not how they are achieved

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions and realizations at the same time

- Mathematically rigorous definition of what a “good” abstraction is
- Should not only hold in isolation but should preserve security under composition.
- Should preserve essentially arbitrary security properties

- Abstractions should match the intuition for the requirements in mind.
- Abstractions should be based on the functionality of the protocol, not on its structure.
- Good abstractions for many of useful protocol (classes) should exist!

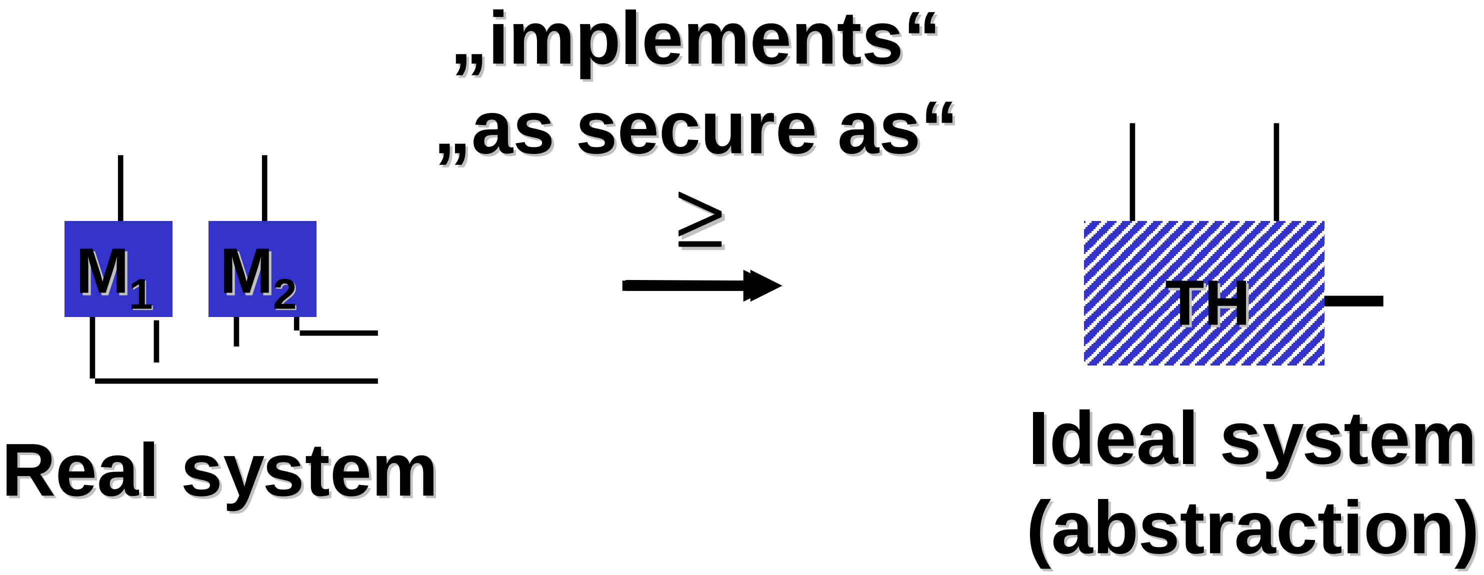
What do we need for soundly abstracting?

- **Precise system model that permits all “realistic” attacks.**
- **Capable of reasoning about abstractions and realizations at the same time**

- Mathematically rigorous definition of what a “good” abstraction is
- Should not only hold in isolation but should preserve security under composition.
- Should preserve essentially arbitrary security properties

- **Abstractions should match the intuition for the requirements in mind.**
- **Abstractions should be based on the functionality of the protocol, not on its structure.**
- **Good abstractions for many of useful protocol (classes) should exist!**

Idea: Define Security relative to an ideal task



„implements“

„as secure as“

$\geq$

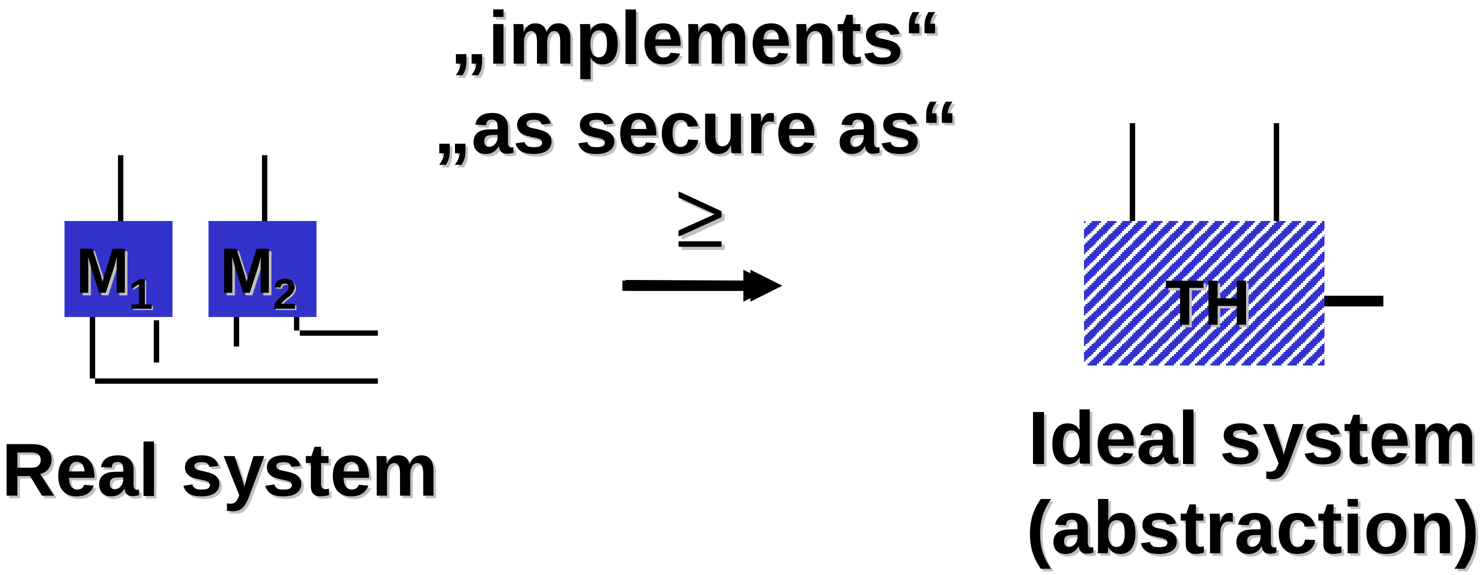
Real system

Ideal system (abstraction)

How to define that? What does “every attack” mean? “successfully converted”?

What are good ideal systems? What about concrete security properties, e.g., integrity or secrecy?

Idea: Define Security relative to an ideal task



„implements“

„as secure as“

≥

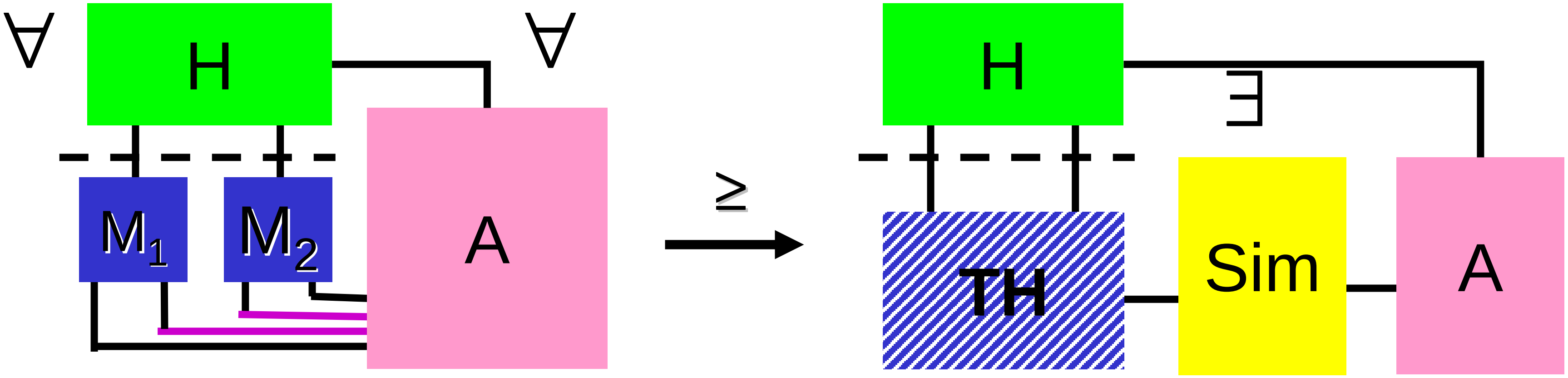
Real system

Ideal system (abstraction)

How to define that? What does “every attack” mean? “successfully converted”?

What are good ideal systems? What about concrete security properties, e.g., integrity or secrecy?

Reactive Simulatability – Top-Level



Real system **Ideal system**

$$\text{view}_{\text{real}}(H) \approx \text{view}_{\text{ideal}}(H)$$

Indistinguishability of random variables

What do we need for soundly abstracting?

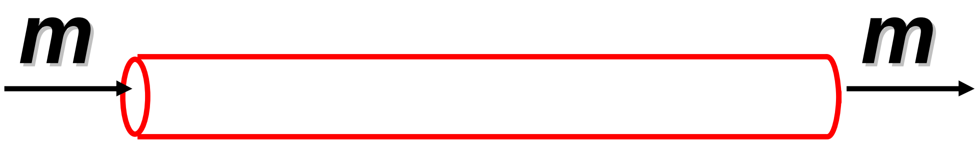
- **Precise system model that permits all “realistic” attacks.**
- **Capable for reasoning about abstractions and realizations at the same time**

- **Mathematically rigorous definition of what a “good” abstraction is**
- **Should not only hold in isolation but should preserve security under composition.**
- **Should preserve essentially arbitrary security properties**

- **Abstractions should match the intuition for the requirements in mind.**
- **Abstractions should be based on the functionality of the protocol, not on its structure.**
- **Good abstractions for many of useful protocol (classes) should exist!**

Naive Approach

E.g., secure channel



Not so easy, e.g.:

- Who-to-whom and length leak.
- No availability
- Ok that error probability etc. omitted?

What Abstractions are good at

+ **Well-defined protocol languages**
+ **Tool-support […]**

– **No cryptographic semantics**
  - **Often term algebras: $D_x(E_x(E_x(m)))$ [DY81]**
  - **“Initial semantics”: No other equations**

– **No techniques for larger modules**

Cryptographic Definitions

+ Precise, proofs possible

– Long and error-prone
  - Adversary
  - Active attacks
  - Error probabilities, computational restrictions

Example: Encryption, passive

$\forall A_1, A_2 \in PPT$:

$P(b^* = b ::$	(Attacker success)
$(sk, pk) \leftarrow gen(k);$	(Keys)
$(m_0, m_1, v) \leftarrow A_1(k, pk);$	(Message choice)
$b \in_R \{0, 1\};$	
$c := enc(pk, m_b);$	(Encrypt)
$b^* \leftarrow A_2(v, c)\ )$	(Guess)
$\leq 1/2 + 1/\mathrm{poly}(k)$	(Negligible)

The Reactive Simulatability Framework Overview

The Reactive Simulatability Framework

- Precise system model **allowing cryptographic and abstract operations**
- Reactive simulatability **with composition theorem**
- Preservation theorems **for security properties**
- **Concrete** pairs of idealizations and secure realizations
- Sound symbolic abstractions **(Dolev-Yao models) that are suitable for tool support**
- Sound security proofs of security protocols**: NSL, Otway-Rees, iKP, etc.**
- Detailed Proofs **(Poly-time, cryptographic bisimulations with static information flow analysis, …)**

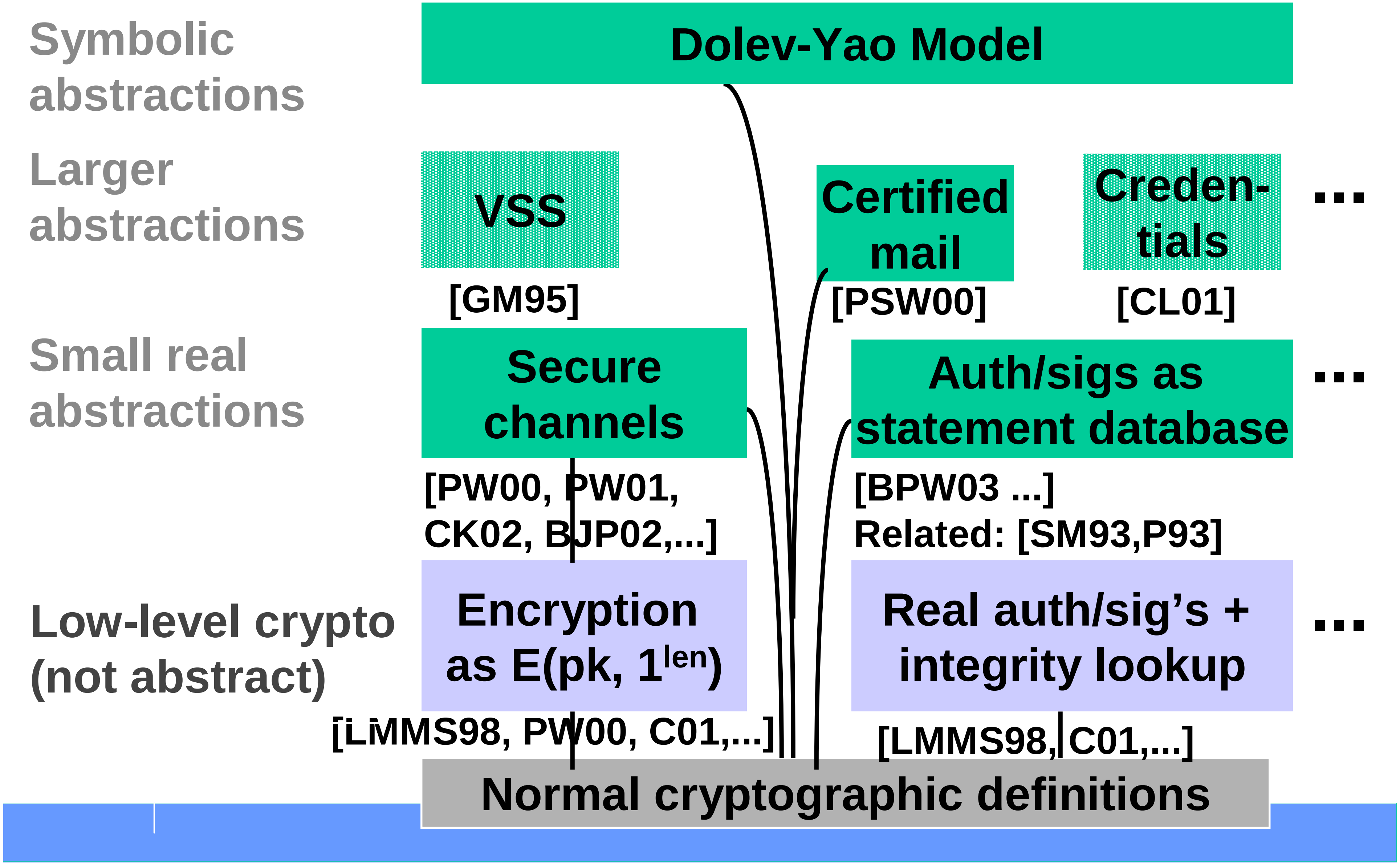
What do we need for soundly abstracting?

- **Precise system model that permits all "realistic" attacks.**
- **Capable for reasoning about abstractions and realizations at the same time**

- **Mathematically rigorous definition of what a "good" abstraction is**
- **Should not only hold in isolation but should preserve security under composition.**
- **Should preserve essentially arbitrary security properties**

- **Abstractions should match the intuition for the requirements in mind.**
- **Abstractions should be based on the functionality of the protocol, not on its structure.**
- **Good abstractions for many of useful protocol (classes) should exist!**

Cryptographic Idealization Layers



Symbolic abstractions

Dolev-Yao Model

Larger abstractions

VSS [GM95]

Certified mail [PSW00]

Creden-tials [CL01]

...

Small real abstractions

Secure channels [PW00, PW01, CK02, BJP02,...]

Auth/sigs as statement database [BPW03 ...] Related: [SM93,P93]

...

Low-level crypto (not abstract)

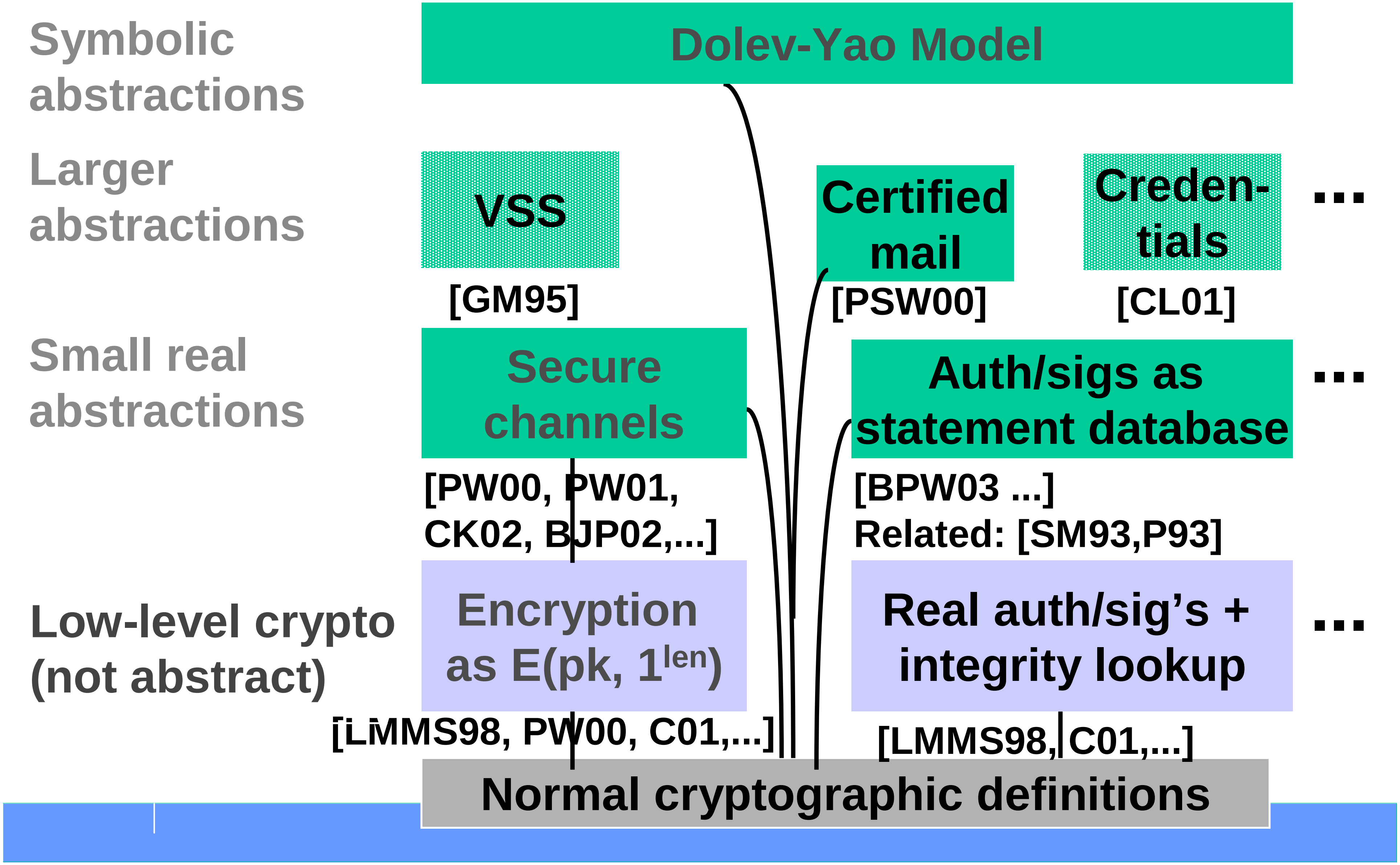
Encryption as E(pk, 1^{len}) [LMMS98, PW00, C01,...]

Real auth/sig's + integrity lookup [LMMS98, C01,...]

...

Normal cryptographic definitions

Cryptographic Idealization Layers



Symbolic abstractions

Dolev-Yao Model

Larger abstractions

VSS

[GM95]

Certified mail

[PSW00]

Creden-tials

[CL01]

...

Small real abstractions

Secure channels

[PW00, PW01, CK02, BJP02,...]

Auth/sigs as statement database

[BPW03 ...]
Related: [SM93,P93]

...

Low-level crypto (not abstract)

Encryption as E(pk, 1^{len})

[LMMS98, PW00, C01,...]

Real auth/sig's + integrity lookup

[LMMS98, C01,...]

...

Normal cryptographic definitions

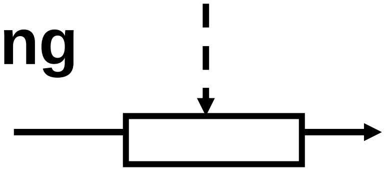
The Reactive Simulatability Framework

- Still today: Precise system model allowing cryptographic and abstract operations
- Reactive simulatability with composition theorem
- Preservation theorems for security properties
- Concrete pairs of idealizations and secure realizations
- Sound symbolic abstractions (Dolev-Yao models) that are suitable for tool support
- Sound security proofs of security protocols: NSL, Otway-Rees, iKP, etc.
- Detailed Proofs (Poly-time, cryptographic bisimulations with static information flow analysis, …)

Definitions Bottom-up (board)

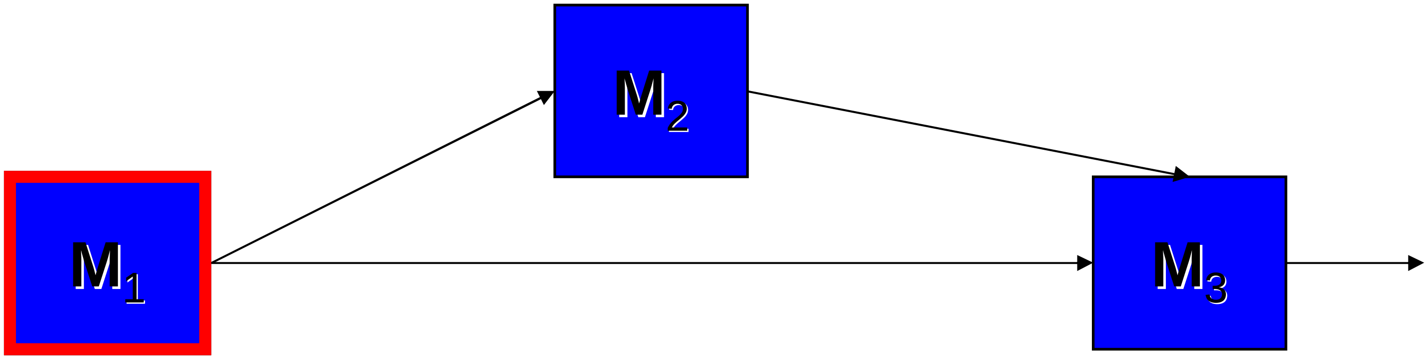
1. General Model:

- Collections of probabilistic I/O automata
  - connections via “ports”
- Turing machine realization (realistic)
- Timing
  - Asynchronous: Distributed scheduling via clock ports
  - Older Synchronous variant: Clk: Subrounds → P(M*)



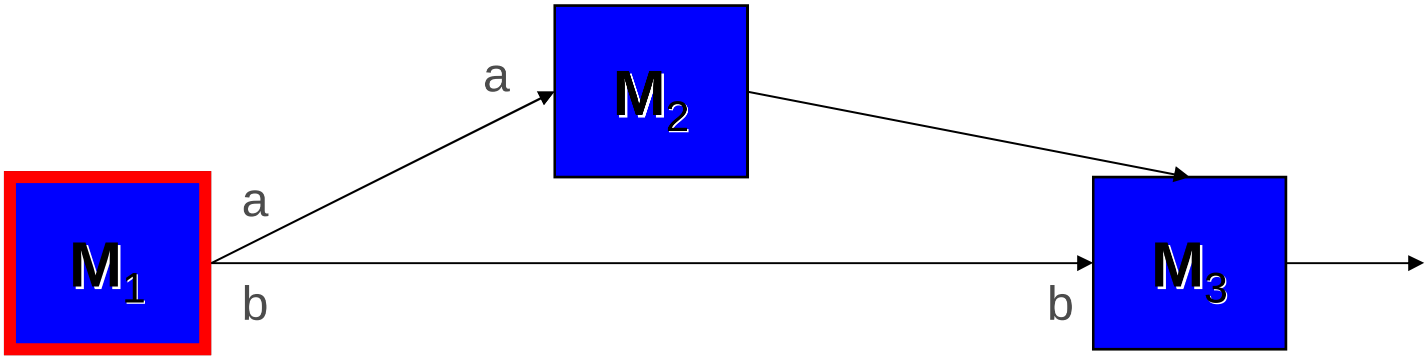
Defining Executions

- (Extended) Probabilistic I/O Automata
- Automata communicate via ports p!, p?, (p !)
- Runs defined for collections of automata:
- First Synchronous:
  - Clocking scheme, e.g., {1} {1,2} {3}



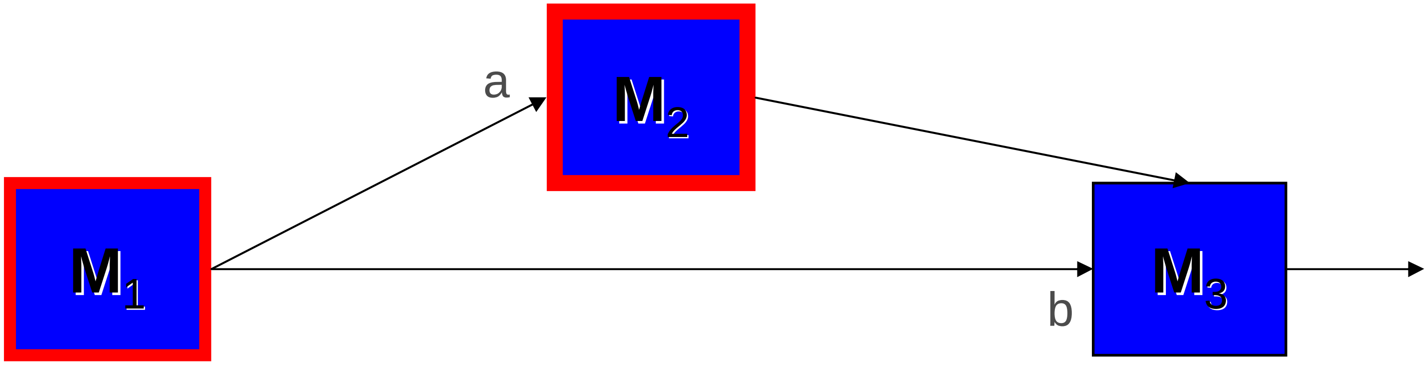
Defining Executions

- **(Extended) Probabilistic I/O Automata**
- **Automata communicate via ports p!, p?, (p !)**
- **Runs defined for collections of automata:**
- **First Synchronous:**
  - **Clocking scheme, e.g., {1} {1,2} {3}**



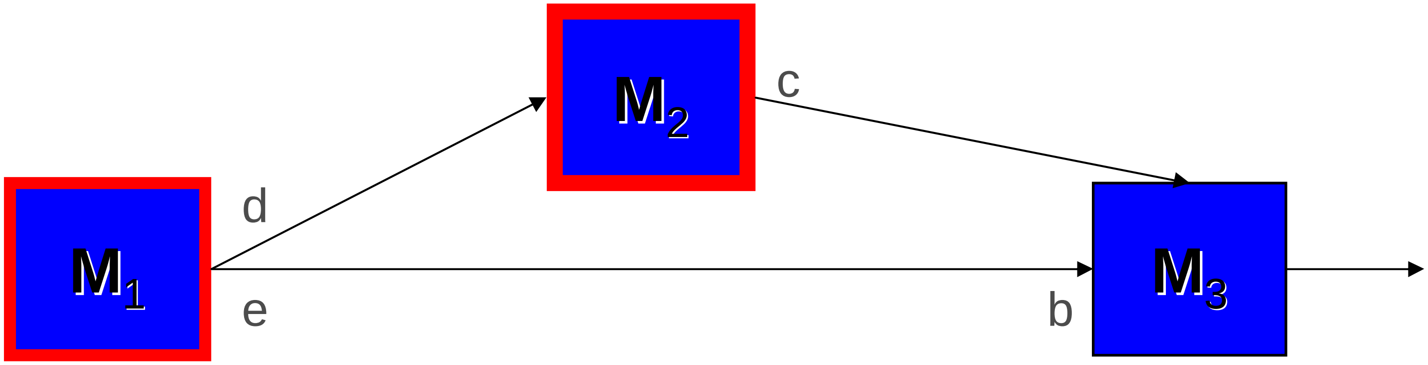
Defining Executions

- (Extended) Probabilistic I/O Automata
- Automata communicate via ports p!, p?, (p !)
- Runs defined for collections of automata:
- First Synchronous:
  - Clocking scheme, e.g., {1} {1,2} {3}



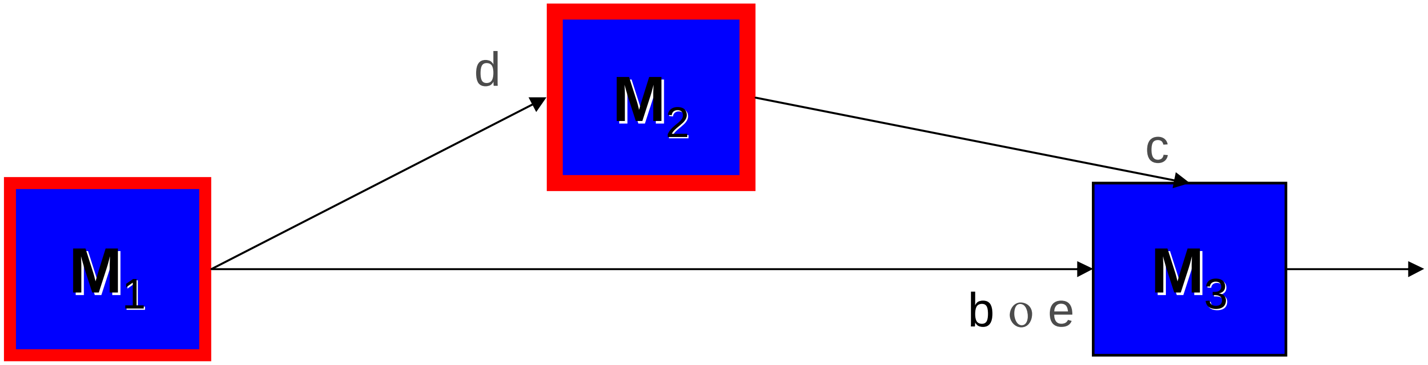
Defining Executions

- **(Extended) Probabilistic I/O Automata**
- **Automata communicate via ports p!, p?, (p !)**
- **Runs defined for collections of automata:**
- **First Synchronous:**
  - **Clocking scheme, e.g., {1} {1,2} {3}**



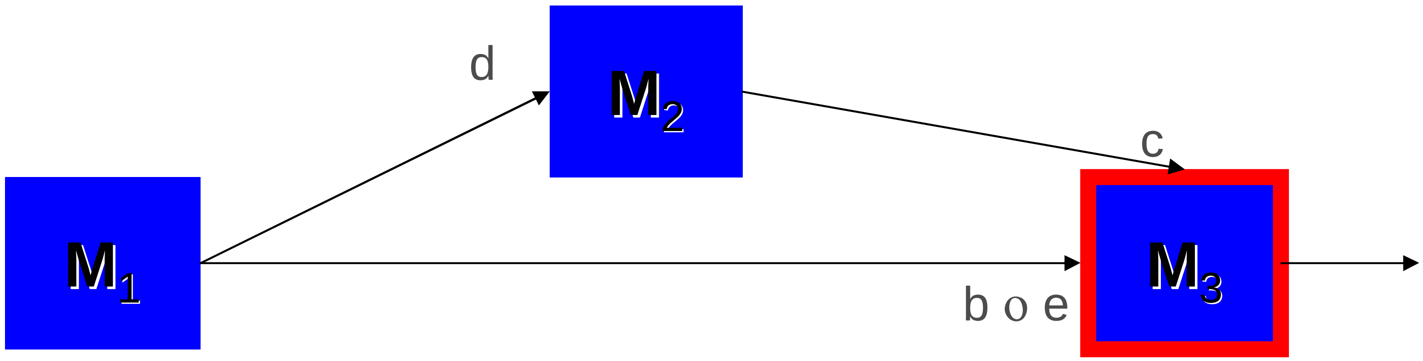
Defining Executions

- **(Extended) Probabilistic I/O Automata**
- **Automata communicate via ports p!, p?, (p !)**
- **Runs defined for collections of automata:**
- **First Synchronous:**
  - **Clocking scheme, e.g., {1} {1,2} {3}**



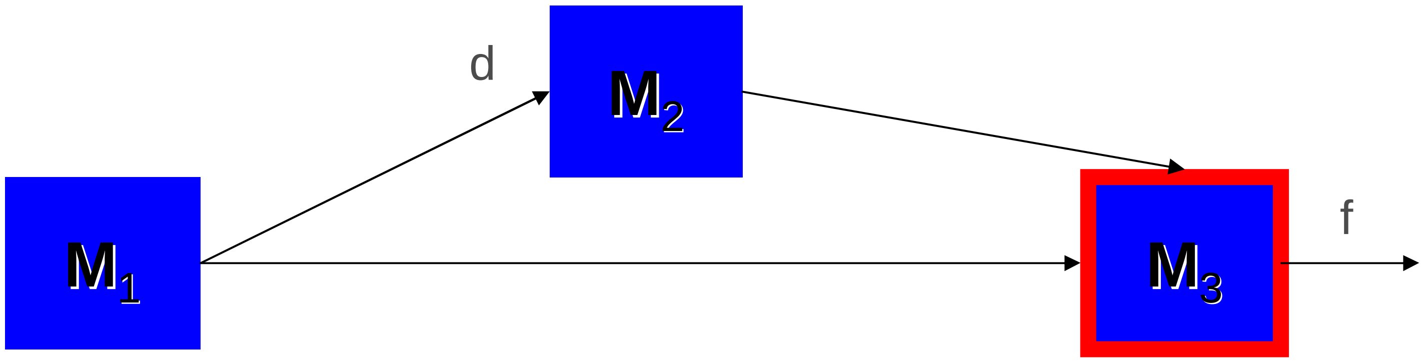
Defining Executions

- (Extended) Probabilistic I/O Automata
- Automata communicate via ports p!, p?, (p !)
- Runs defined for collections of automata:
- First Synchronous:
  - Clocking scheme, e.g., {1} {1,2} {3}



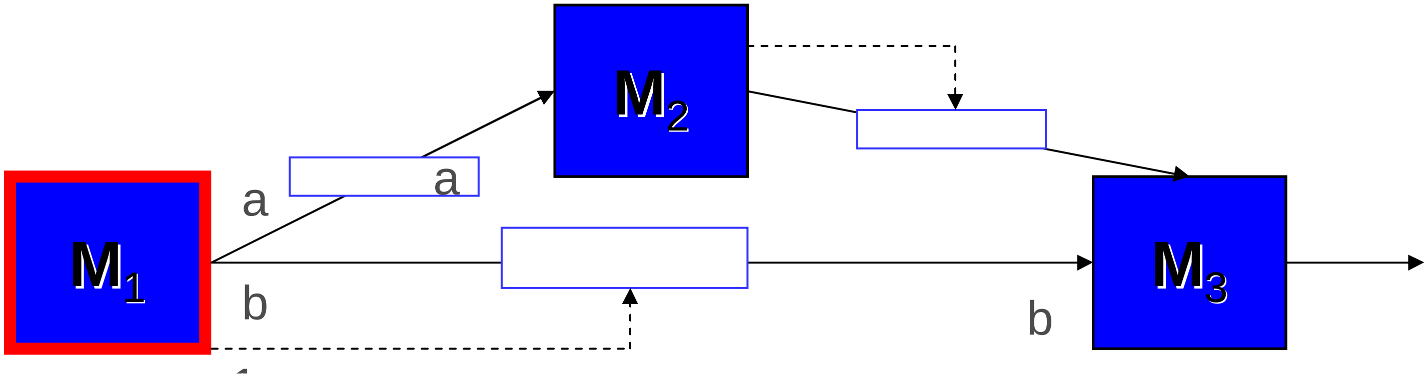
Defining Executions

- (Extended) Probabilistic I/O Automata
- Automata communicate via ports p!, p?, (p !)
- Runs defined for collections of automata:
- First Synchronous:
  - Clocking scheme, e.g., {1} {1,2} {3}



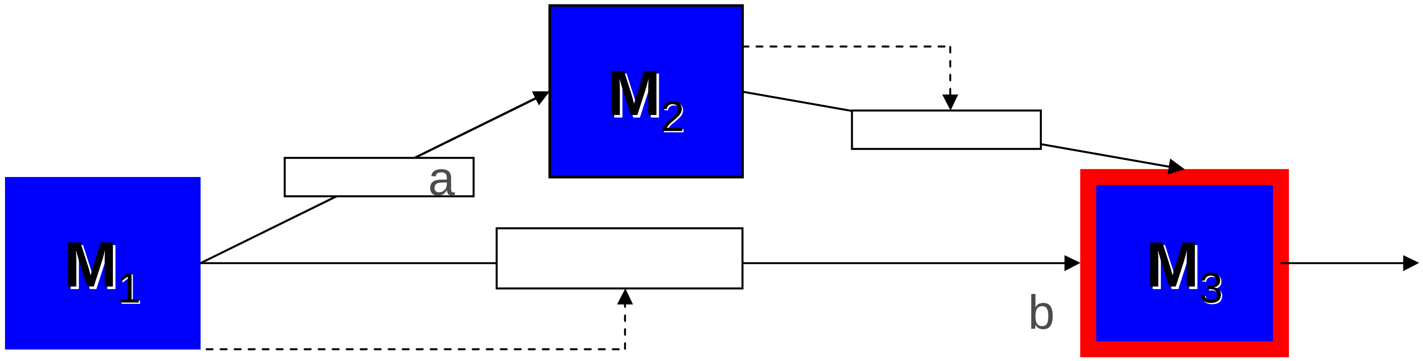
General Model

- **Probabilistic I/O Automata [SL95]**
- **Automata communicate via ports p!, p?, (p !)**
- **Runs defined for collections of automata:**
- **Now Asynchronous:**
    - **Only one machine active, sequential scheduling, master scheduler**



General Model

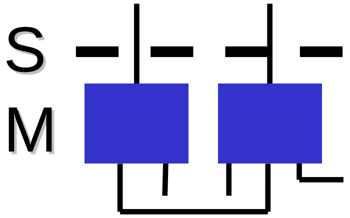
- **Probabilistic I/O Automata [SL95]**
- **Automata communicate via ports p!, p?, (p !)**
- **Runs defined for collections of automata:**
- **Now Asynchronous:**
  - **Only one machine active, sequential scheduling, master scheduler**



Definitions Bottom-up

2. Security-Specific System Model:

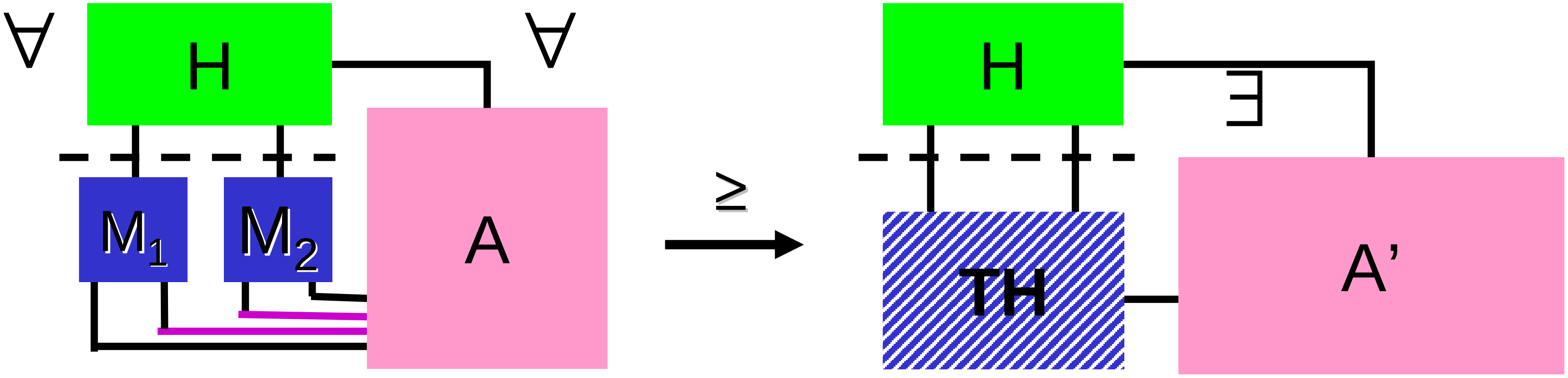
- **Structure: (M, S) with $S \subseteq$ Ports(M)**
  “service ports”



- **Configurations: (M, S, H, A)**

Reactive Simulatability (“as secure as”)

Soundness: Reactive Simulatability



Real system **Ideal system**

$$\text{view}_{\text{real}}(H) \;\approx\; \text{view}_{\text{ideal}}(H)$$

Indistinguishability of random variables

Outlook for Tomorrow

- **Precise system model allowing cryptographic and abstract operations**
- **Reactive simulatability with composition theorem**
- **Preservation theorems for security properties**
- **Concrete pairs of idealizations and secure realizations**
- **Sound symbolic abstractions (Dolev-Yao models) that are suitable for tool support**
- **Sound security proofs of security protocols: NSL, Otway-Rees, iKP, etc.**
- **Detailed Proofs (Poly-time, cryptographic bisimulations with static information flow analysis, …)**