Rigorous Model of MAFTIA Concepts

Presented by André Adelsbach
Saarland University, Saarbrücken, Germany

MAFTIA Workshop, Newcastle, Feb 18-19, 2003

WP 6: Goals and Achievements

- Validate results of dependable middleware
  - Formal specification in CSP and verification with FDR
- Provide rigorous general system model which allows for
  - Rigorous definitions of basic concepts
    - Modular systems, synchrony, fault- and attack-models
  - Modular security proofs (composition theorem)
  - Combined use of
    - Complexity-theoretic cryptographic proofs

      \+ Realistic abstractions

      – Not accessible to automatic checking of proofs
    - Formal methods

      – Unrealistic abstractions, esp. of cryptographic primitives

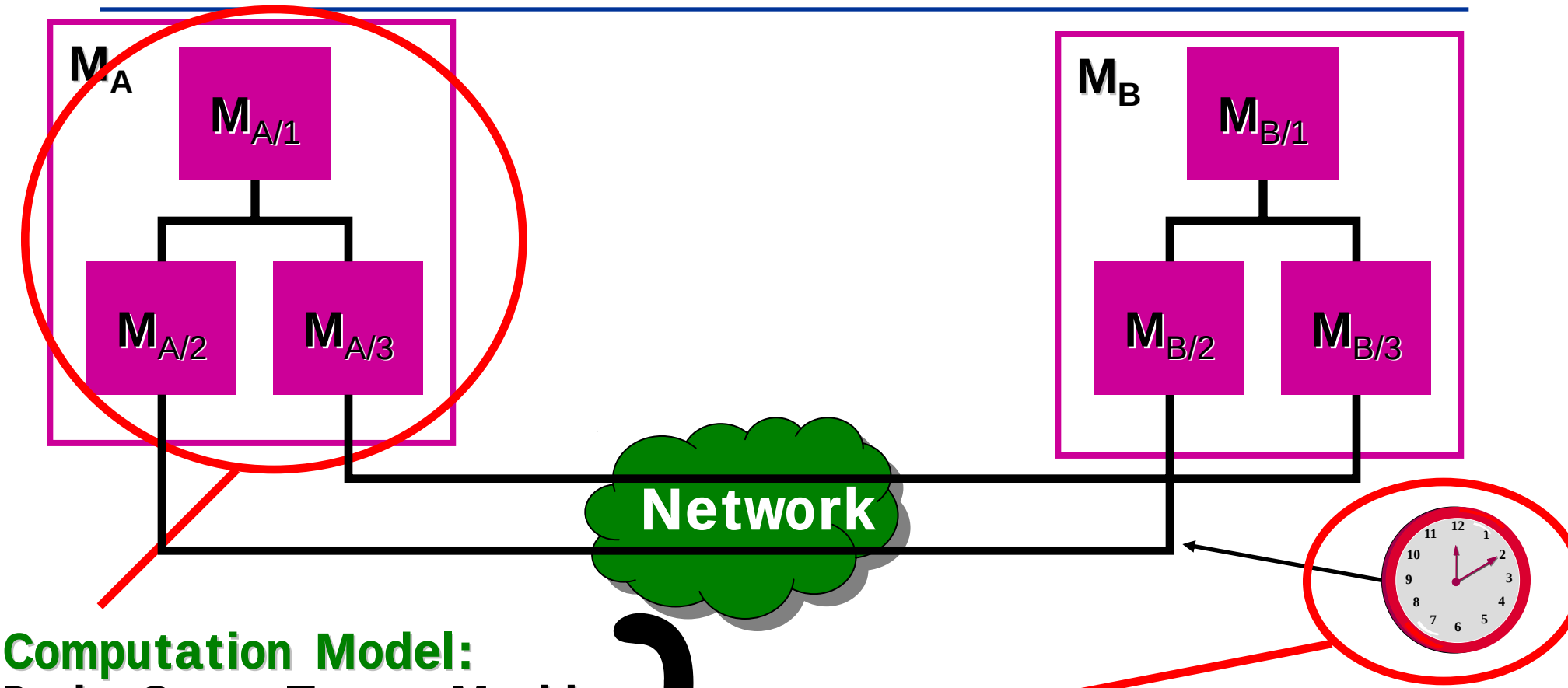
      \+ Tool support

Outline of Talk

- **Rigorous General System Model (D4, D8)**
  - Modular systems, synchrony, faults, errors, failures
  - General security definitions & modular proofs
- **A Rigorous Model of a MAFTIA Primitive (D22)**
  - Key establishment
- **Cautionary Note (D8, D22)**
  - Gap between system model and the real world
  - Impact on real world security when implementing secure systems

Rigorous General System Model

Modelling a Simple MAFTIA System



Computation Model:
Prob. State Trans. Machines

Communication Model:
Synchronous &
Asynchronous Networks

Behaviour of Systems:
Prob. space on runs, i.e., sequ. of tuples (states, in-, outputs)

Modelling Faults, Errors and Failures

Fault: adjudged or hypothesized cause of an error

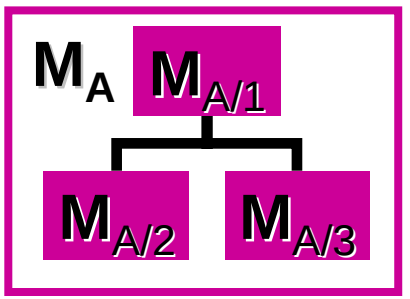
Error: that part of "system state" which may lead to a failure

Failure: delivered service deviates from desired service

Reality

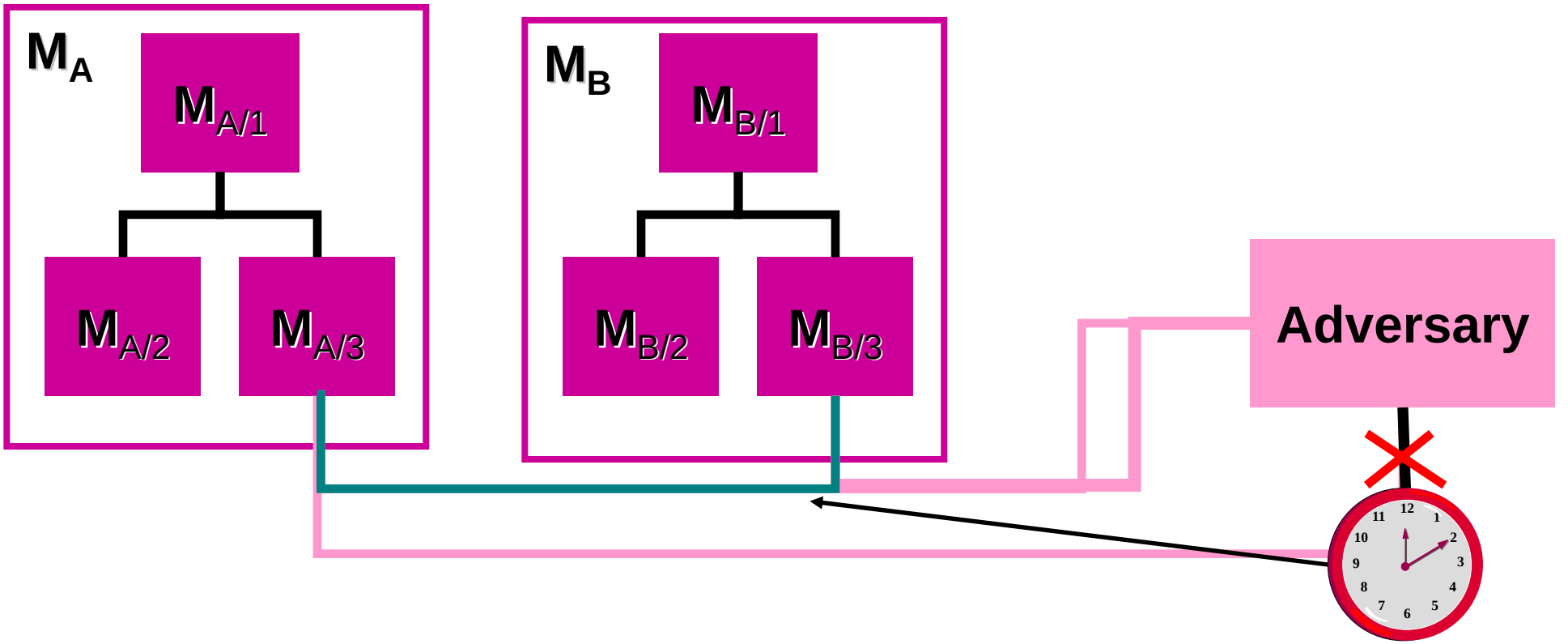
Fault: cosmic radiation
Error: bit-flip in memory
Failure: displays wrong price

System Model



Fault: ???
Error: bit-flip in variable
Failure: outputs wrong price

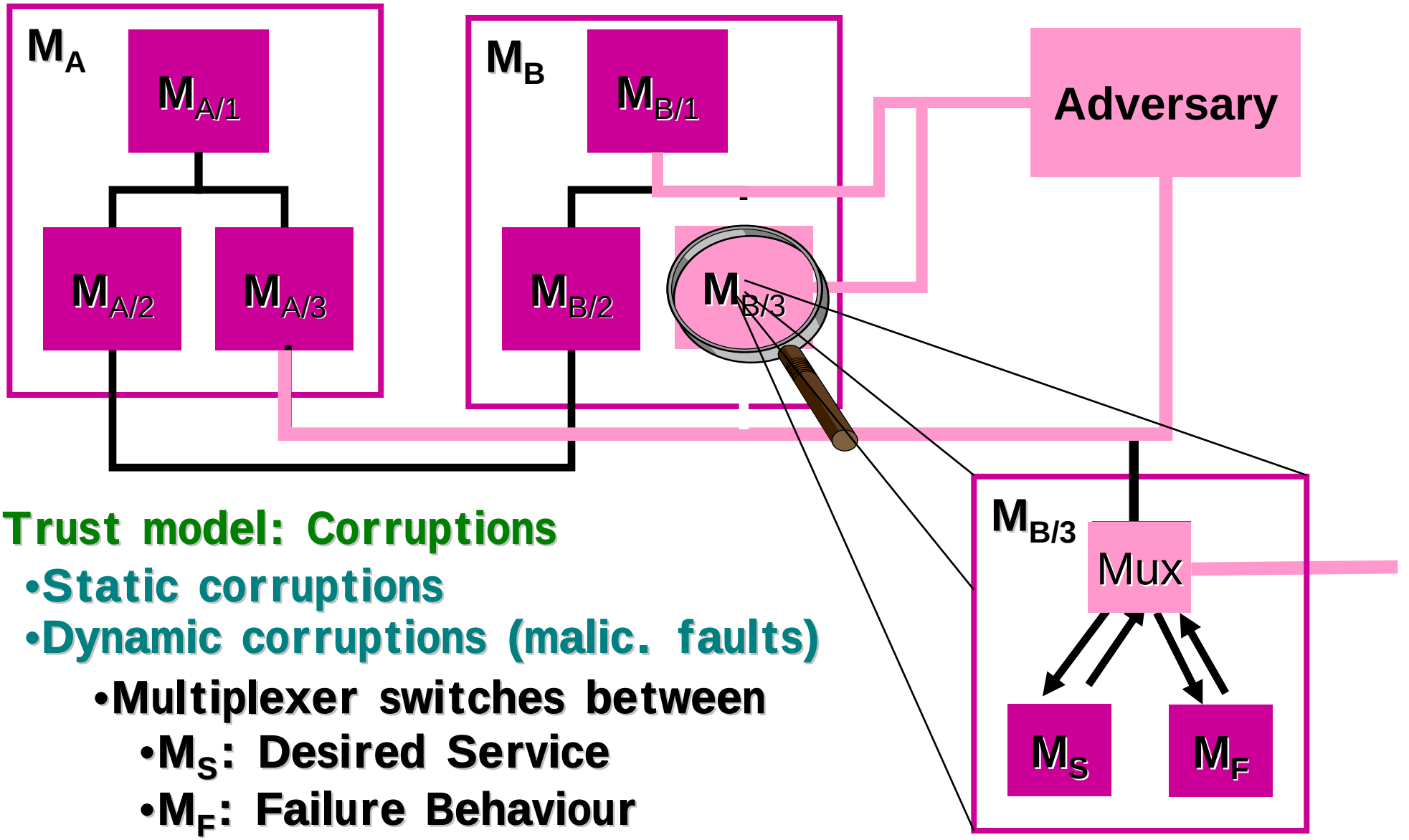
General Attacks



Attacker model: General capabilities !

- Computational power
- Network control (secure, authentic, insecure, reliable)
- Corruption

General Attacks: Corruptions



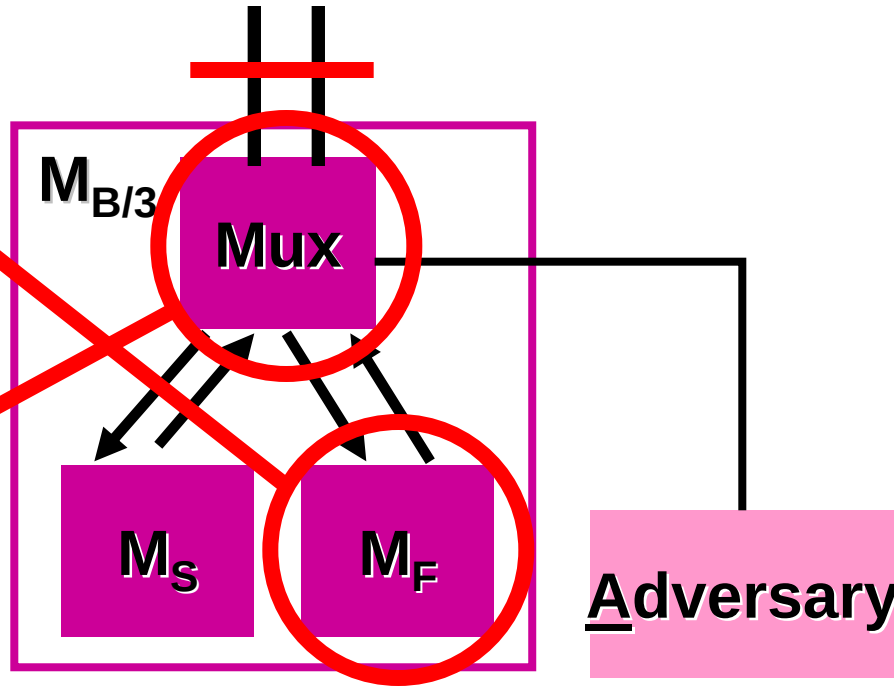
Trust model: Corruptions

- **Static corruptions**
- **Dynamic corruptions (malic. faults)**
  - **Multiplexer switches between**
    - **M_S: Desired Service**
    - **M_F: Failure Behaviour**

Modelling Failures

Controlled vs Uncontrolled

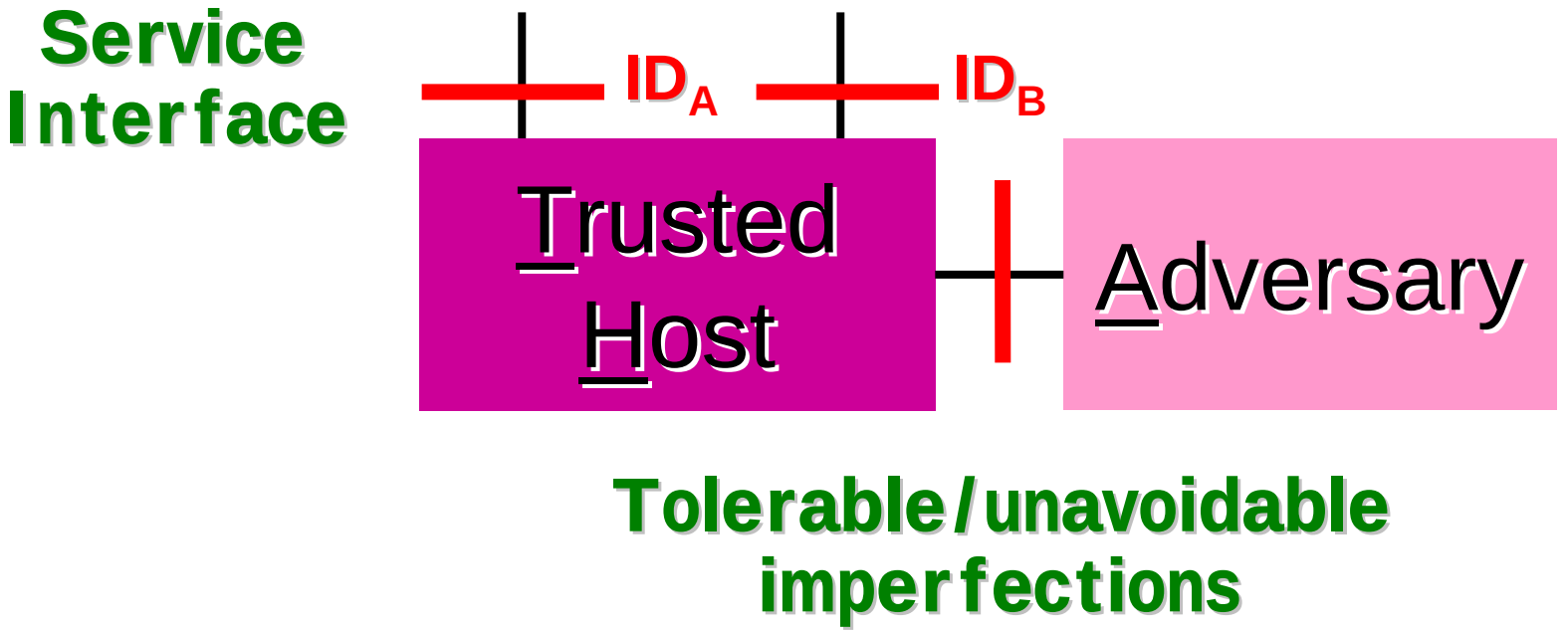
- **Controlled Failures:**
  - degraded service
  - Crash failure
  - …
- **Uncontrolled Failures:**
  - M_F defined to output arbitrary messages



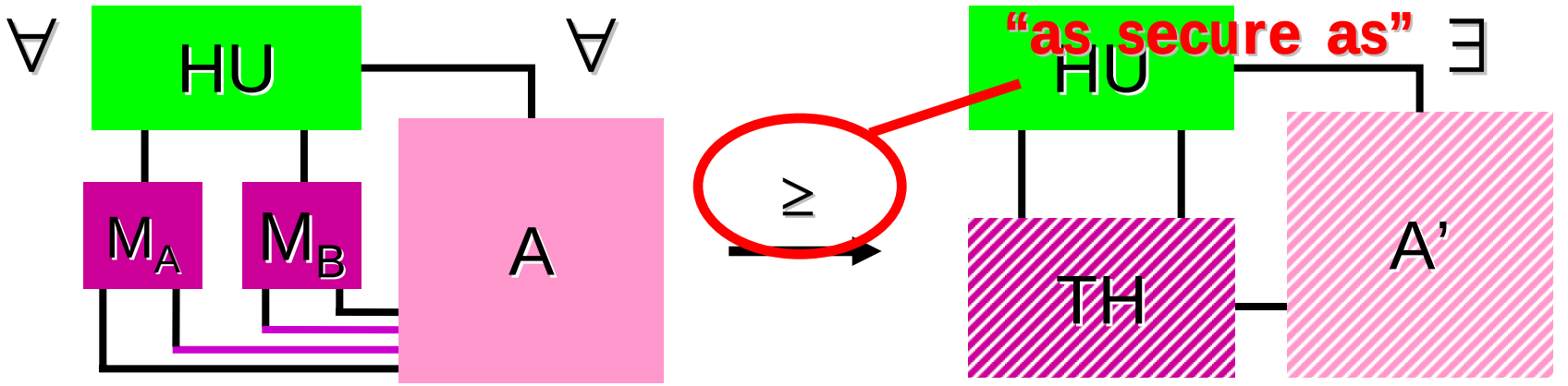
Malicious vs. Non-malicious Faults

- **non-malicious faults:** Mux triggers itself
- **malicious faults:** Mux triggered by adversary machine

General Security Requirements



Security Proof: Simulatability



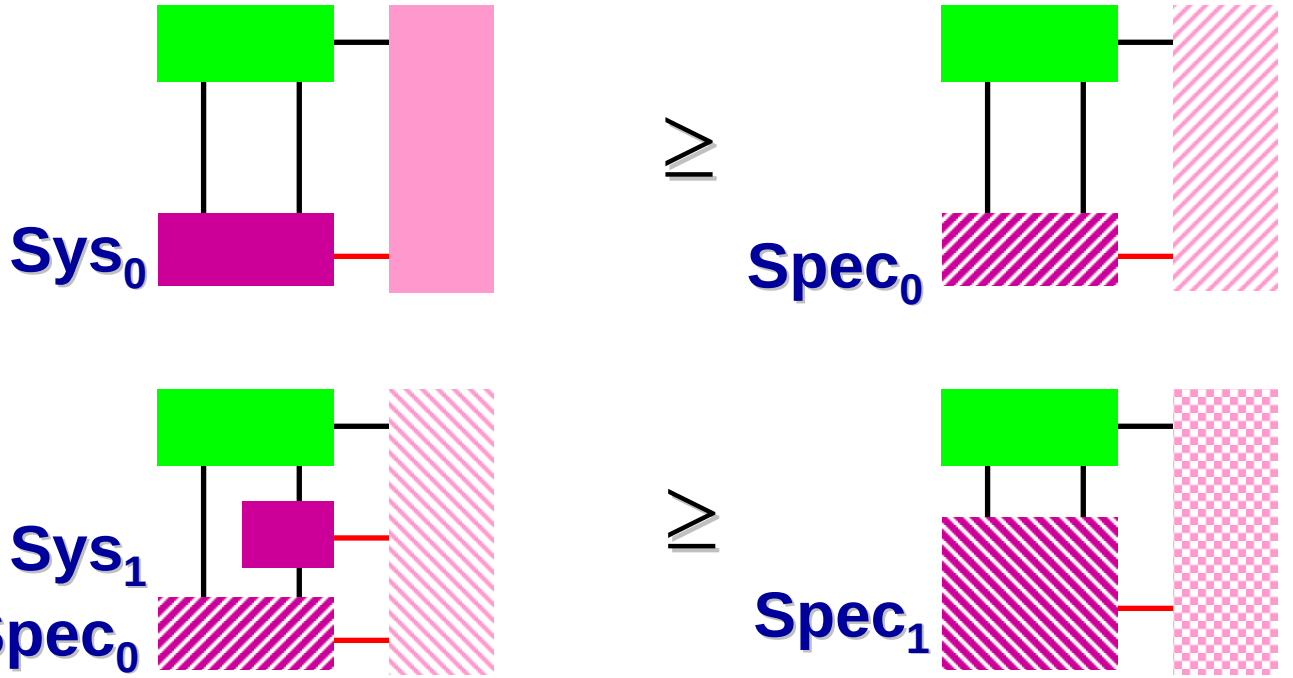
Real System **Ideal System**

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

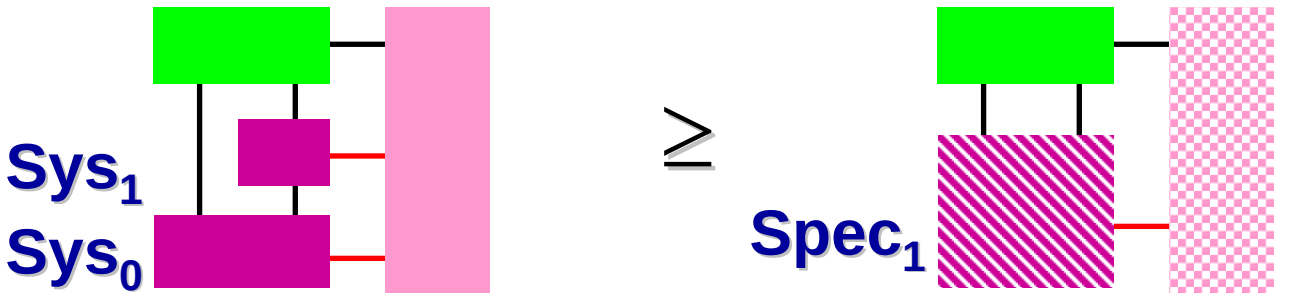
Indistinguishability of induced probability distributions

Composition Theorem

Given:



Composition Theorem ⇒



Example: Key Establishment

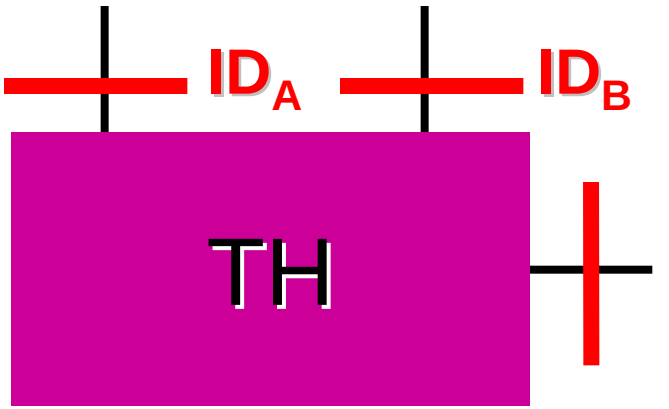
What is Key Establishment (KE) ?

- **KE-Service:**
  - Given long-term keys, two (or more) parties agree on a session key
- **Security Requirements:**
  - Authenticity: Agreement on context (key, identities, session,…)
  - Confidentiality: Key unpredictable to outsider
  - Freshness: Key independent of other keys

KE: Specification of Desired Service

Service Interface

if (ID_A & ID_B ask for session){
    generate random key
    return key to ID_A & ID_B
}



Imperfections

- Leak traffic info
- Let adversary control session establishment

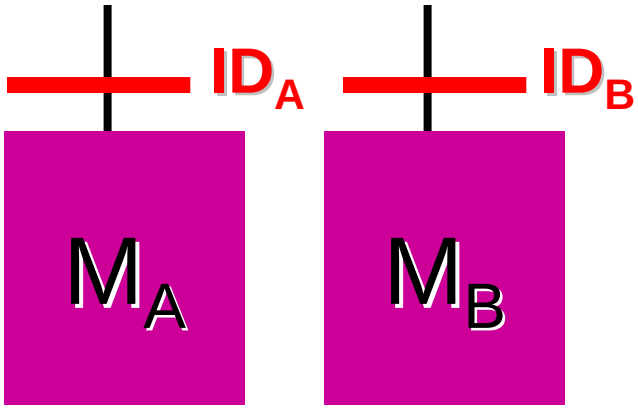
Properties

- Authenticity & freshness
- Key bit-string & every bit unpredictable
- …

KE: Real Protocol

Key-Establishment protocol [STW00]

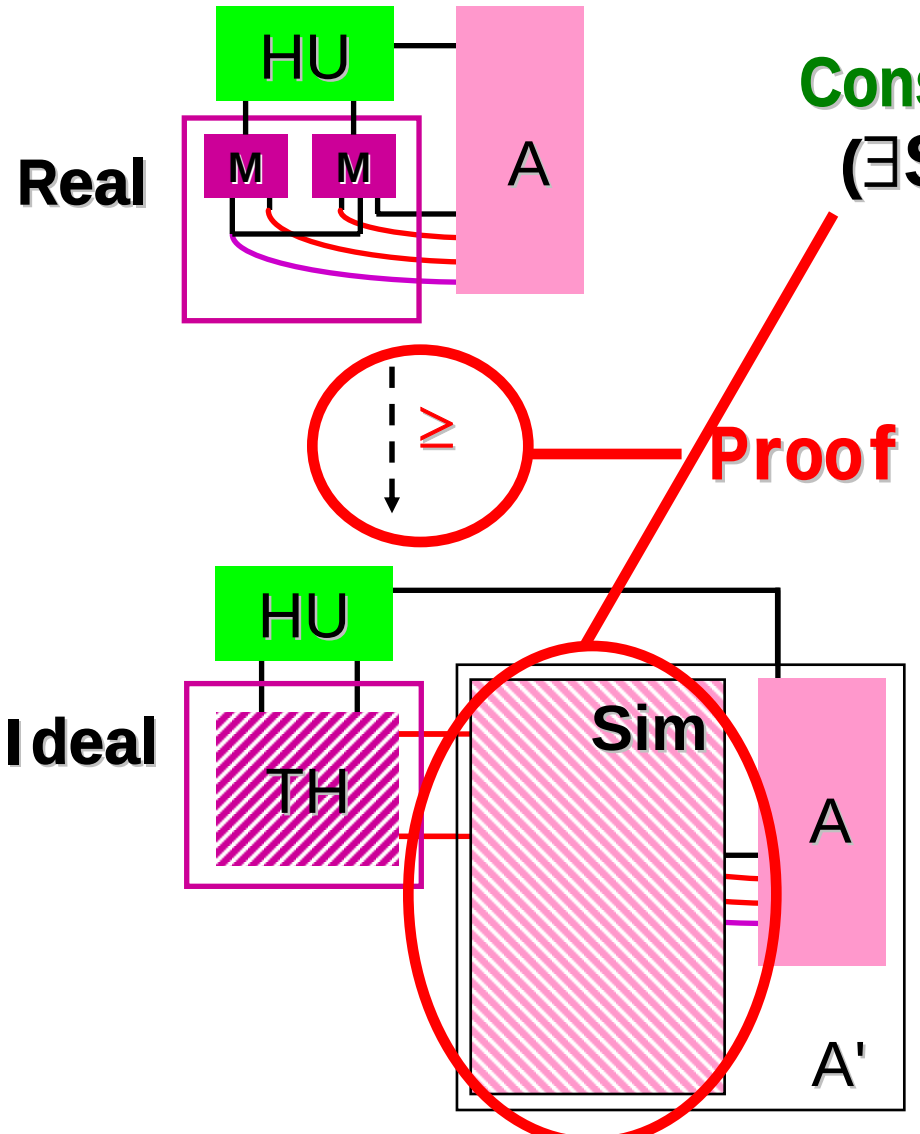
- Provably secure…
- but also practical



Building Blocks

- **Authentication:** Signatures & Identifiers
- **Confidentiality:** Diffie-Hellman key-exchange
- **Adaptive corruption:** Synchronise & erase session-specific secrets

KE: Proof Idea (Goal)

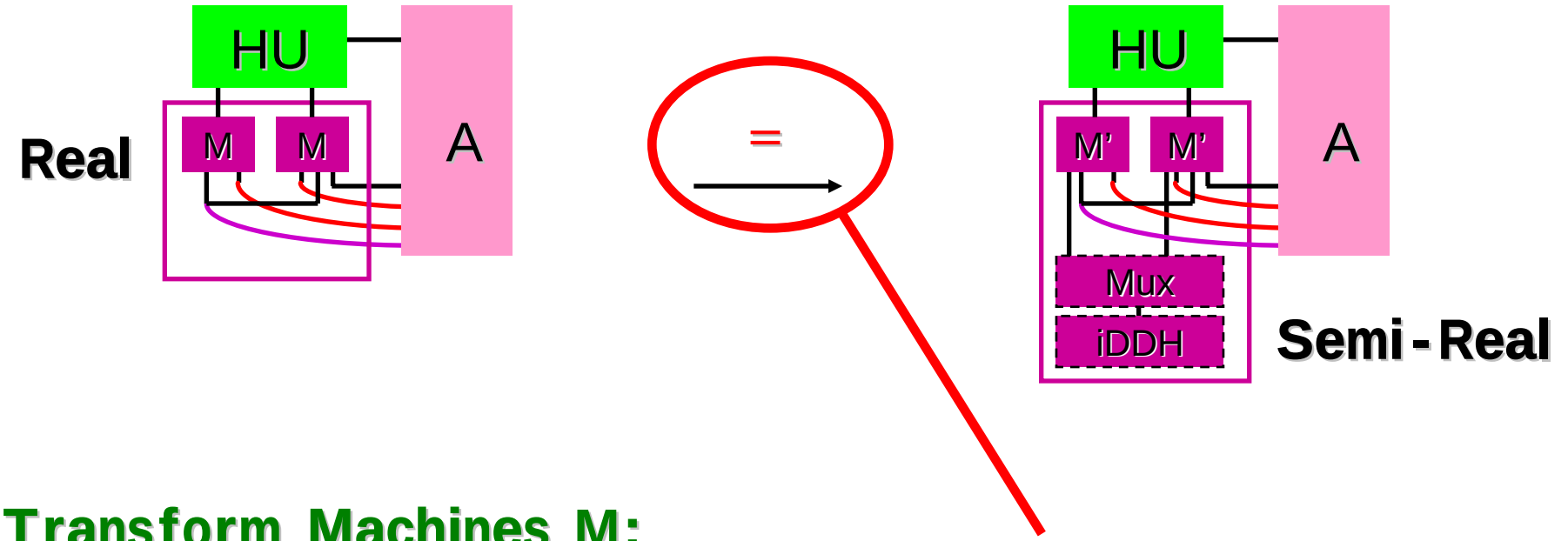


Construct black-box simulator

(∃Sim ∀A ∀H: A' = Sim&A)

Proof in several steps

KE: Proof Idea (Step 1)



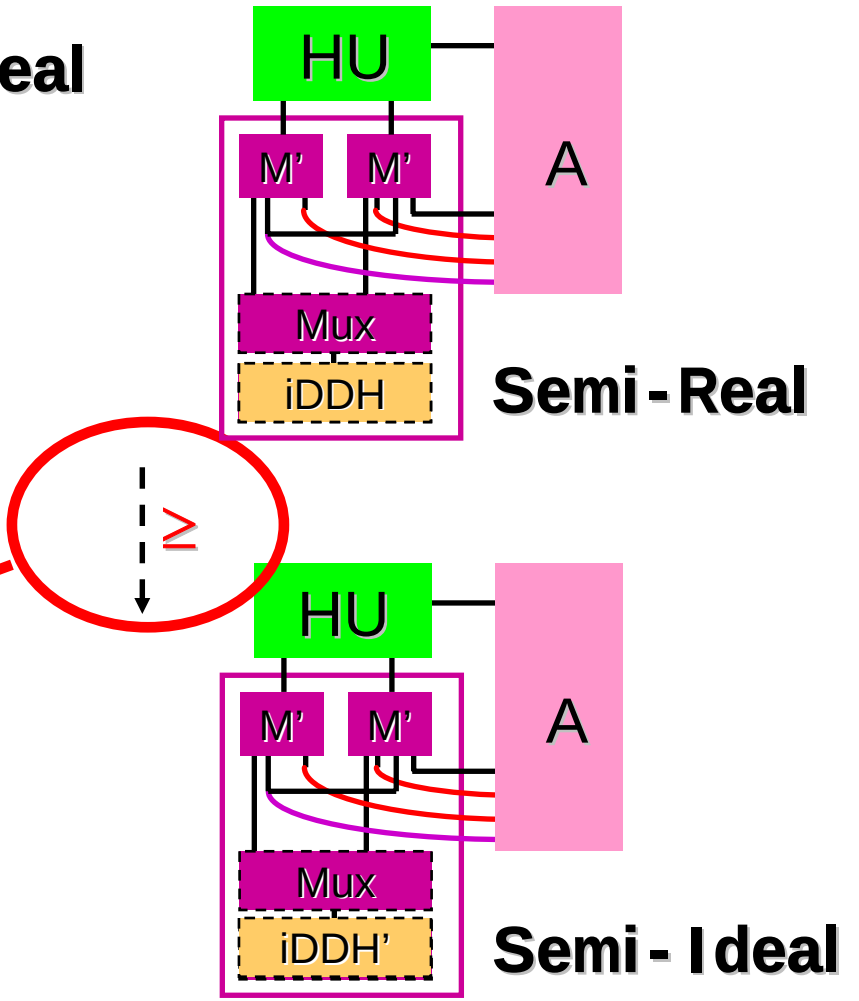
Transform Machines M:

- **iDDH:** Encapsulates cryptographic problem
- **Mux:** Abstract key computation and handle adaptive corruption

Bisimulation (identical probabilistic behavior)

KE: Proof Idea (Step 2)

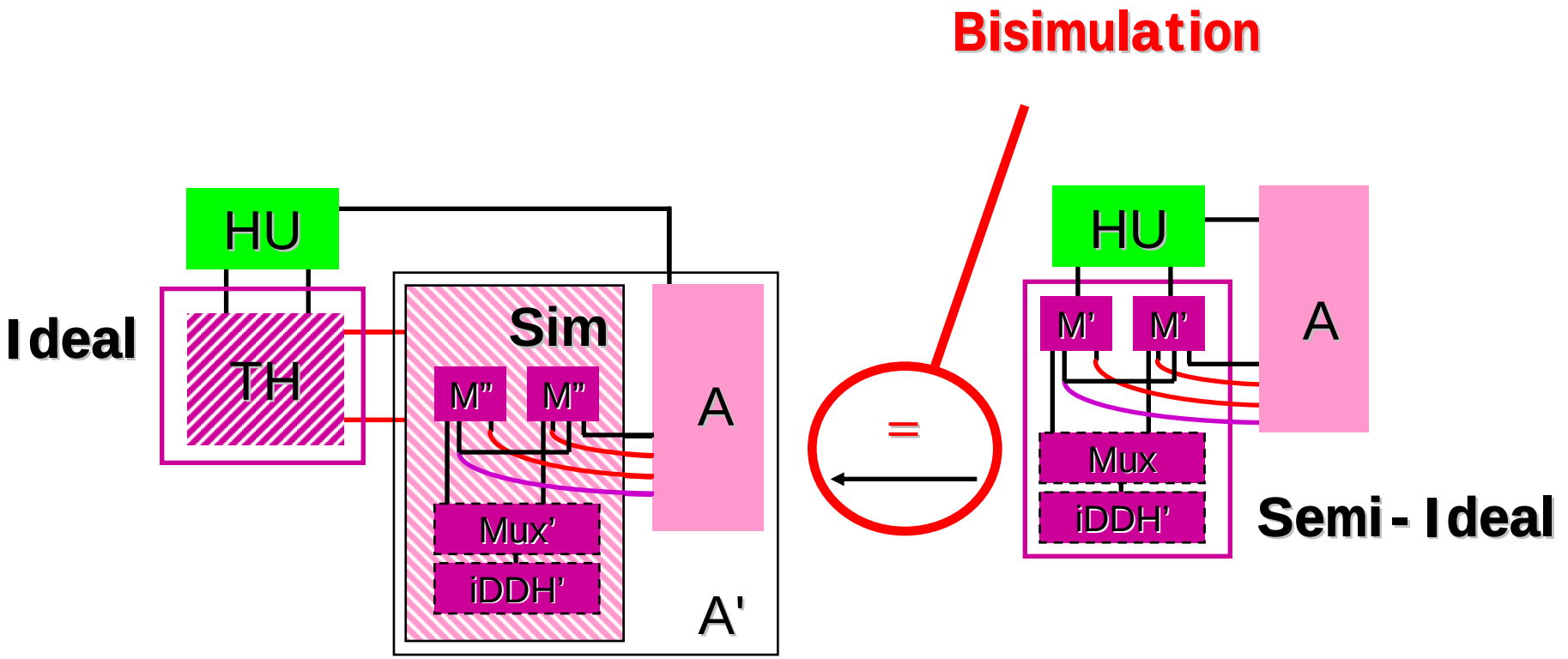
- Replace real (DH) keys by ideal (random) keys (iDDH')



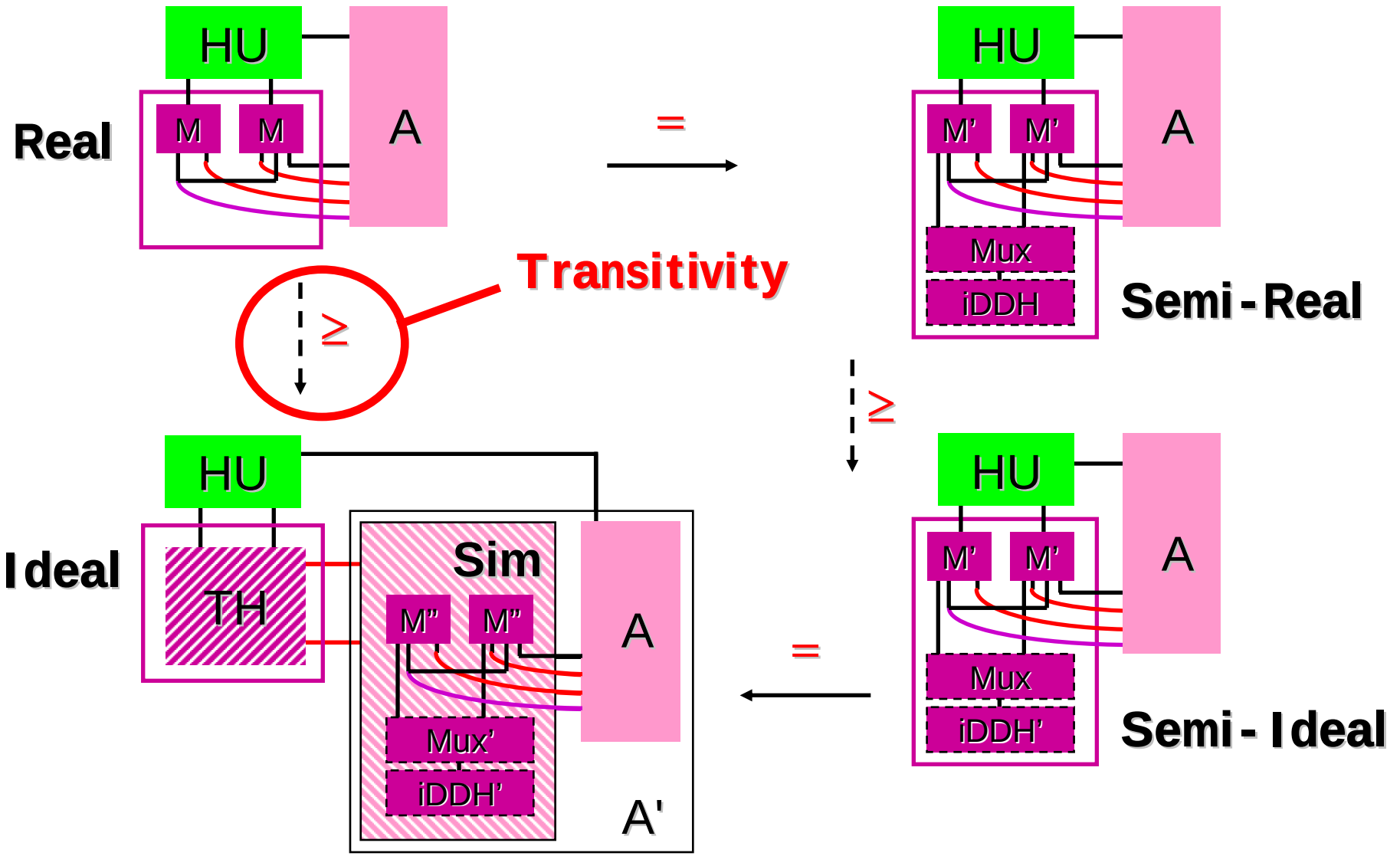
- Prove indistinguishability of iDDH and iDDH'
- Composition theorem

KE: Proof Idea (Step 3)

- Embed semi-ideal system in simulator to translate view of A
- Computation of ideal keys in TH instead of Mux



KE: Proof Idea (Step 4)

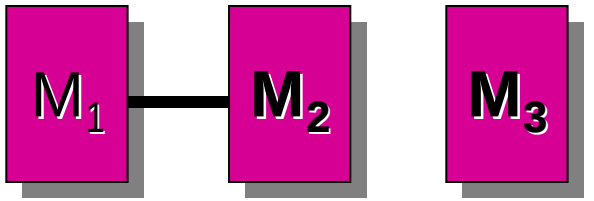


Final “Cautionary” Note

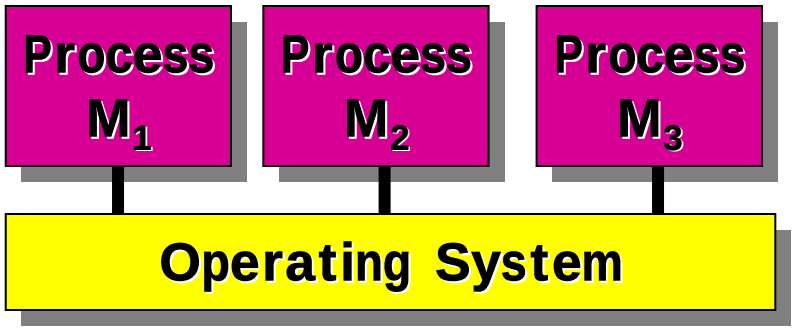
Gap between system model and the real world

Impacts on Real-World Security

Model



Real World



- Access only through interfaces
- Type enforcement
- ....

- No strict separation
- Covert channels
  - Timing channels
  - Storage channels
- No type enforcement
  - Buffer overflows
- ....