ETH zürich

Overcoming Impossibility Results in Composable Security using Interval-Wise Guarantees

Daniel Jost Ueli Maurer

ETH Zurich

Crypto 2020, August 17-21, 2020

D INFK

Motivation: how to best define security?

Defining Security

Game-based security:

- Simple and minimal

- no direct link to real-world executions
- many games
- no composition

Defining Security

Game-based security:

- Simple and minimal

- no direct link to real-world executions
- many games
- no composition

Composable security:

- Guarantees linked to real-world application
- Modularization
- Composition

- More complicated proofs
- Less efficient schemes
- Impossibility results

Defining Security

Game-based security:

- Simple and minimal

- no direct link to real-world executions
- many games
- no composition

Composable security:

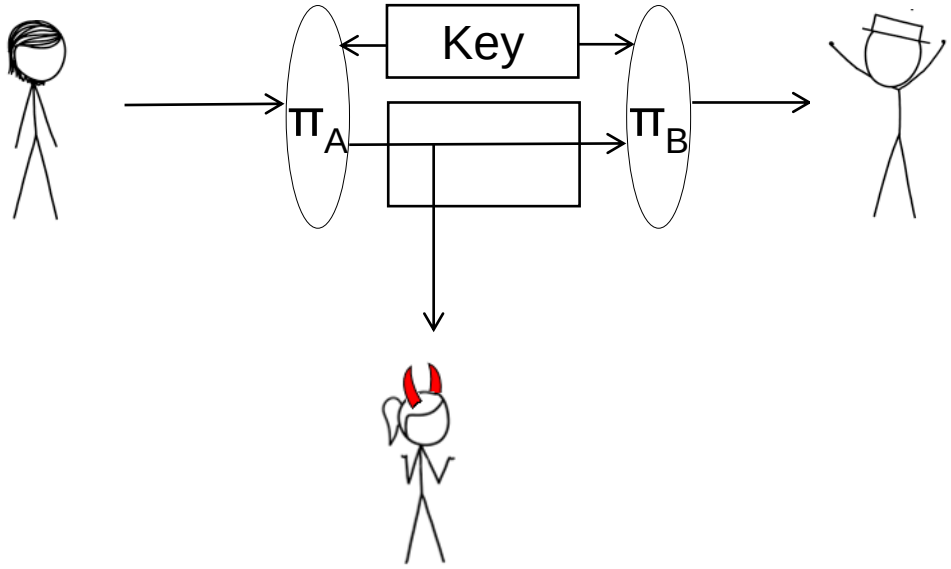
- Guarantees linked to real-world application
- Modularization
- Composition

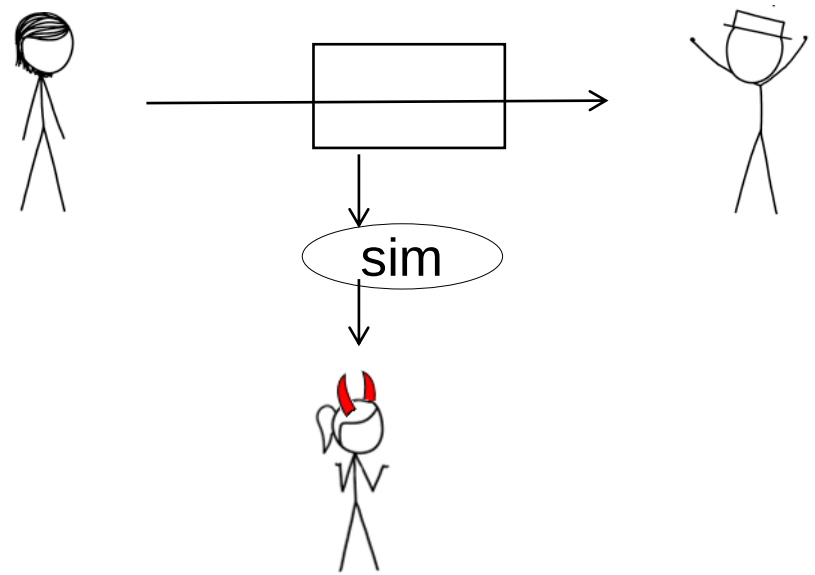
- More complicated proofs
- Less efficient schemes
- Impossibility results

Simulator-commitment problem

The Commitment Problem

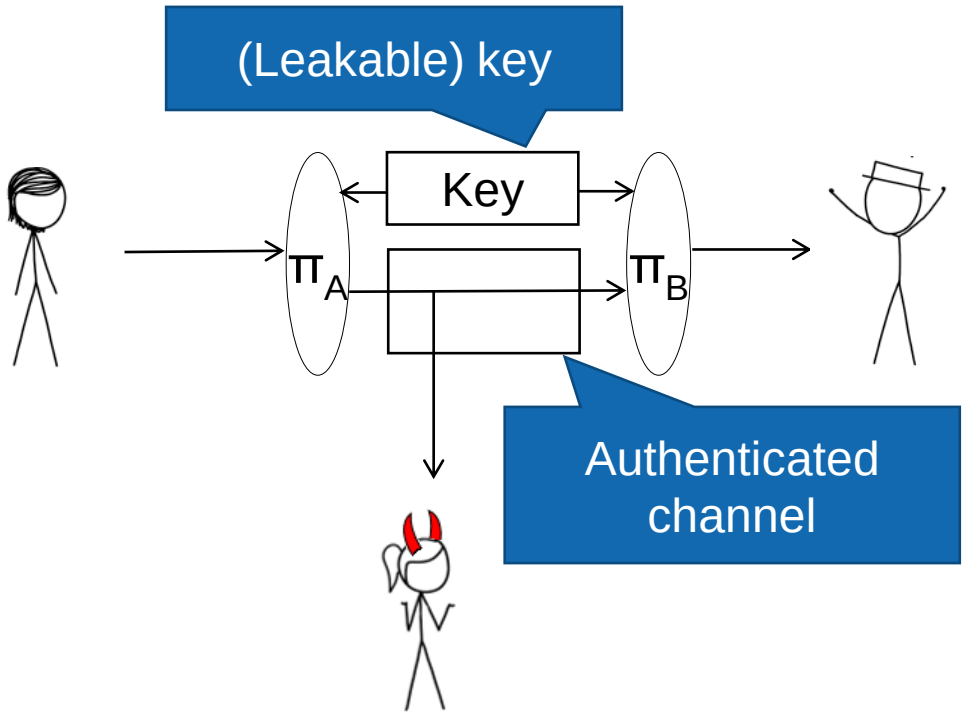
- **Example**:
  - encrypting a message to protect confidentiality
  - where adversaries that can (adaptively) learn parties' state (including keys)

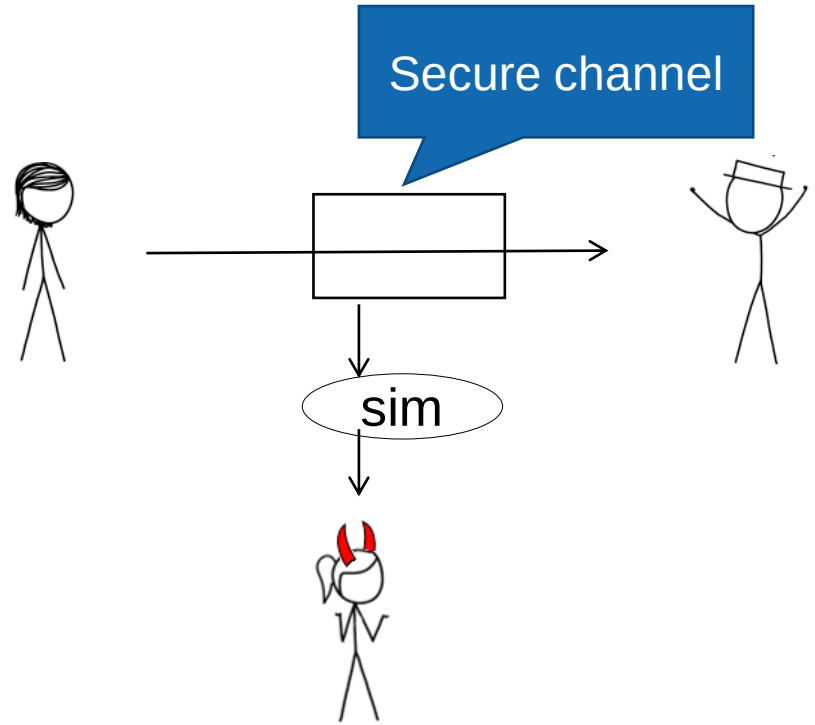




The Commitment Problem

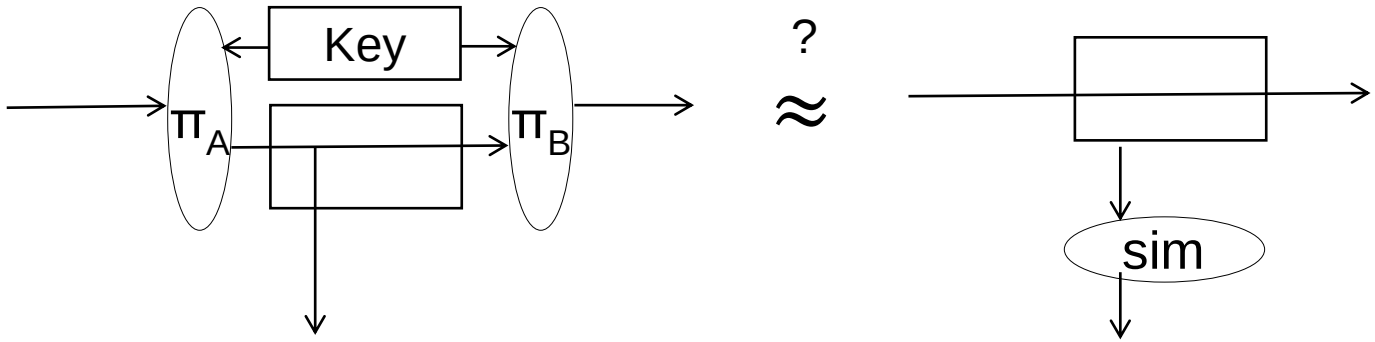
- **Example**:
  - encrypting a message to protect confidentiality
  - where adversaries that can (adaptively) learn parties' state (including keys)





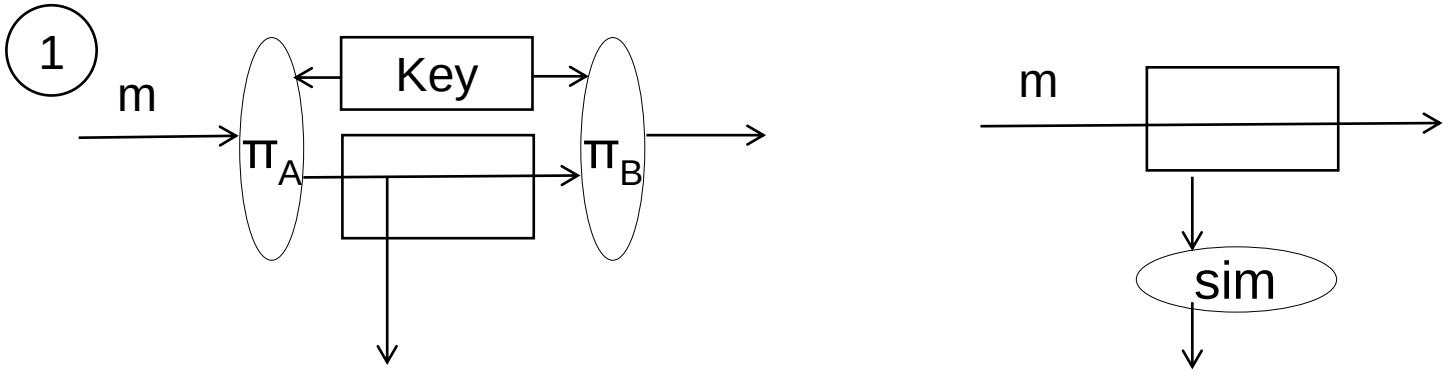
The Commitment Problem

- **Example**:
  - encrypting a message to protect confidentiality
  - where adversaries that can (adaptively) learn parties' state (including keys)



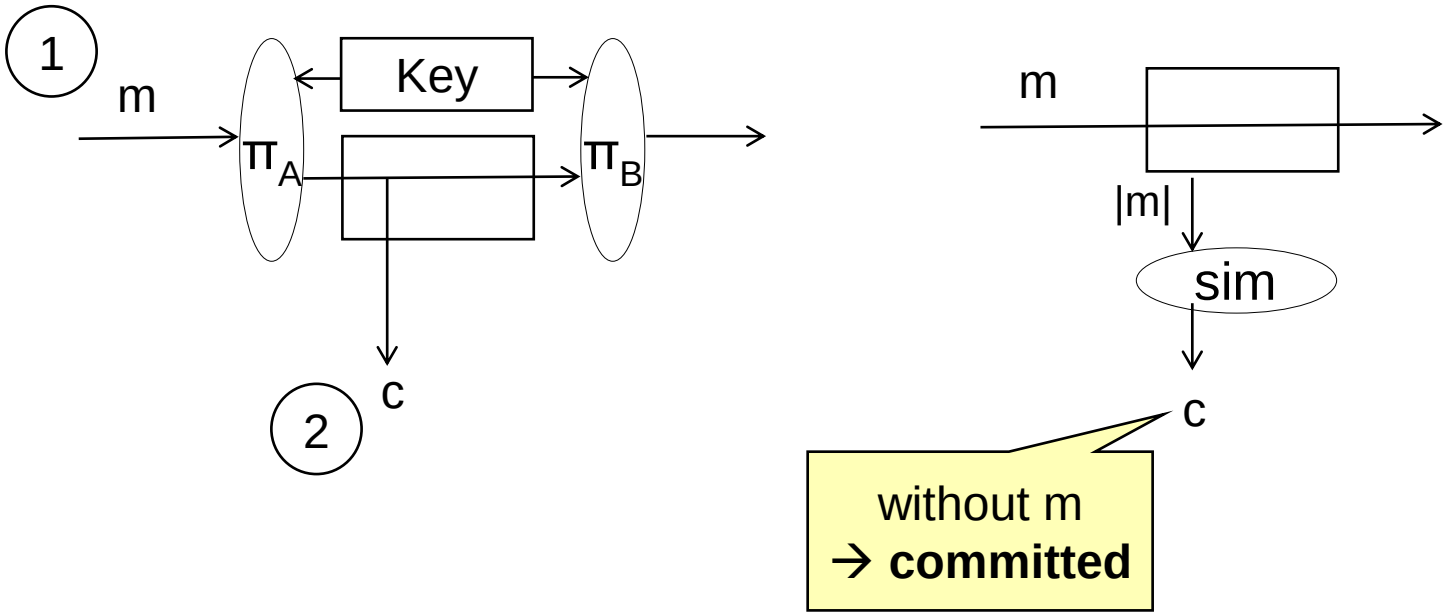
The Commitment Problem

- **Example**:
  - encrypting a message to protect confidentiality
  - where adversaries that can (adaptively) learn parties' state (including keys)



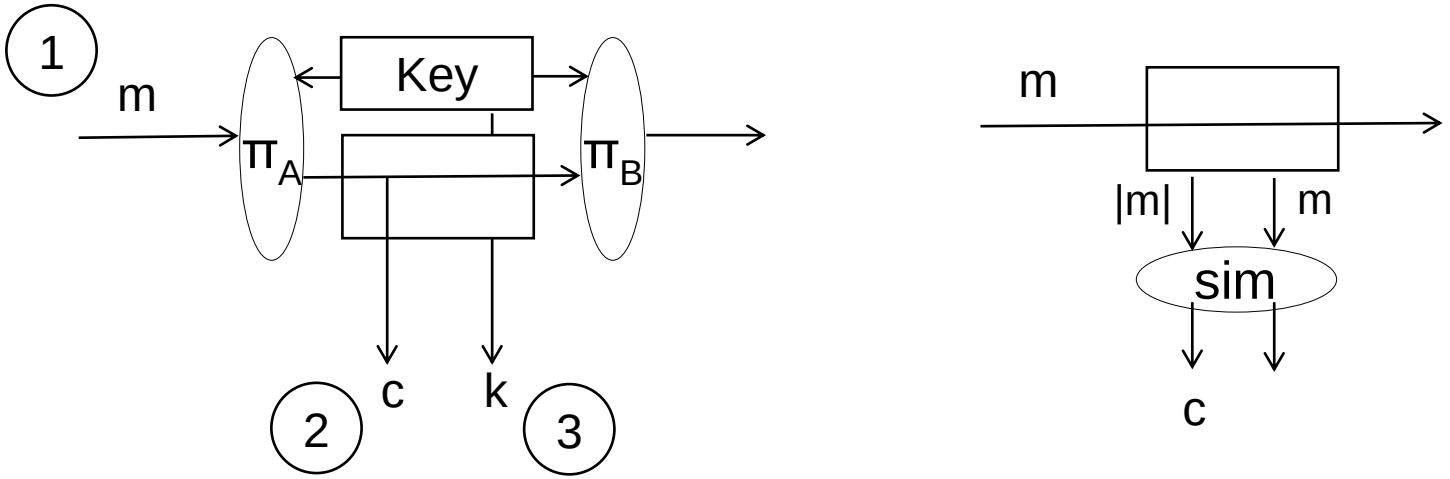
The Commitment Problem

- **Example**:
  - encrypting a message to protect confidentiality
  - where adversaries that can (adaptively) learn parties' state (including keys)



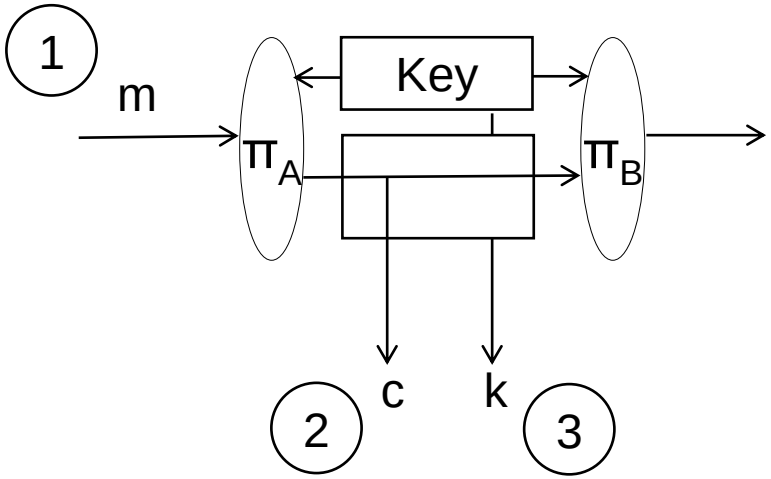
The Commitment Problem

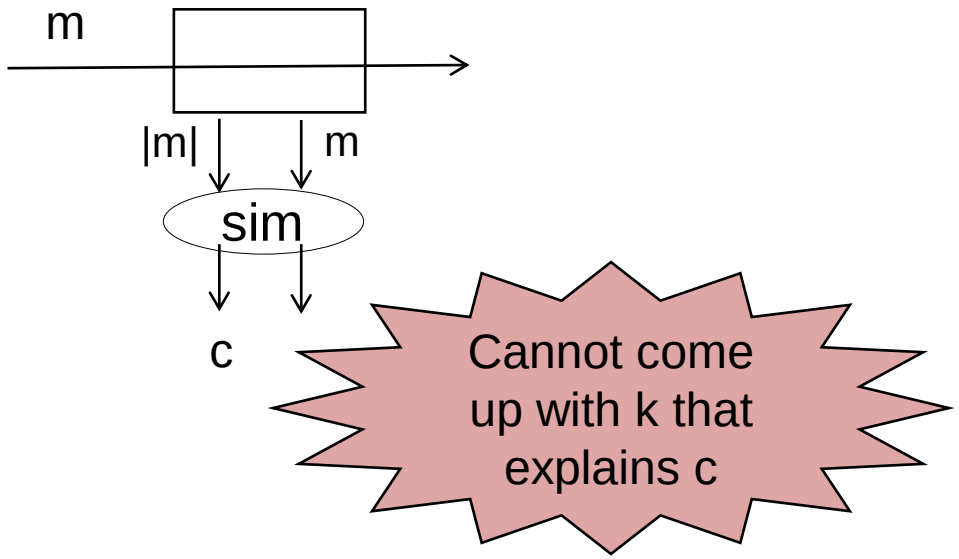
- **Example**:
  - encrypting a message to protect confidentiality
  - where adversaries that can (adaptively) learn parties' state (including keys)



The Commitment Problem

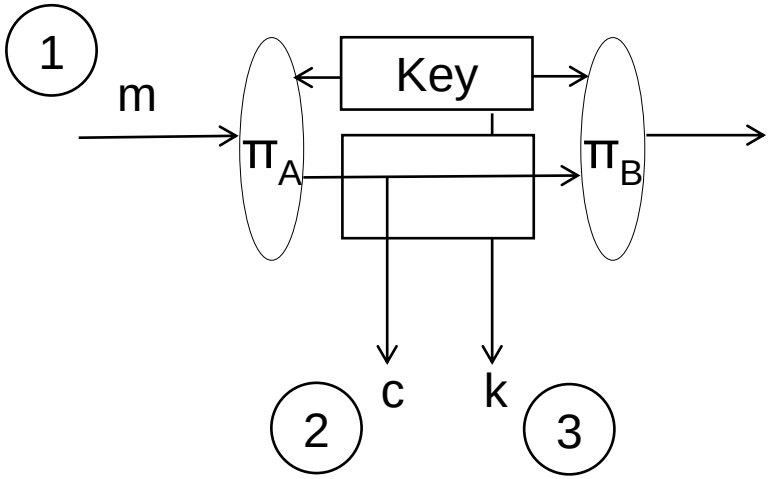
- **Example**:
  - encrypting a message to protect confidentiality
  - where adversaries that can (adaptively) learn parties' state (including keys)

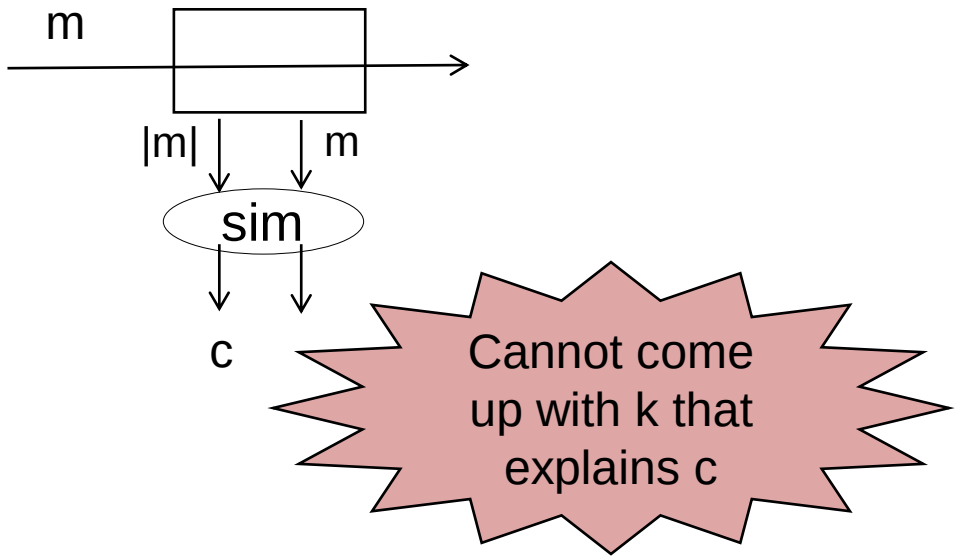




Observation:

- Leaking only the messages length (and the simulator creating a fake ciphertext) is used to formalize that the message remains **confidential until the key leaks**
- But it causes problems to simulate after that event…





The Commitment Problem

- **Existing solutions**:

  – Allowing for superpolynomial simulators

    → Still needs stronger schemes / additional setup

  – Non-information oracles: embedding game-based notions

    → Lack of clear composition rules

Contributions

Idea of this paper: Can we make to separate statements?

- One up to the moment the key leaks (for confidentiality)
- One after the key leaked (about the remaining guarantees)

Goal:

- Express security guarantees of «regular» schemes composably

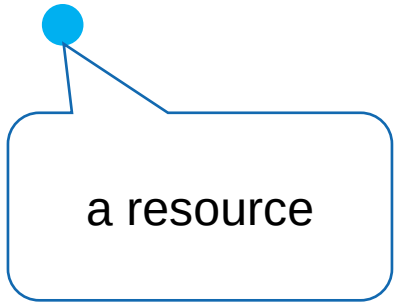
Non-goals:

- Requireing less efficient schemes / additional setup
- Fall back to game-based security

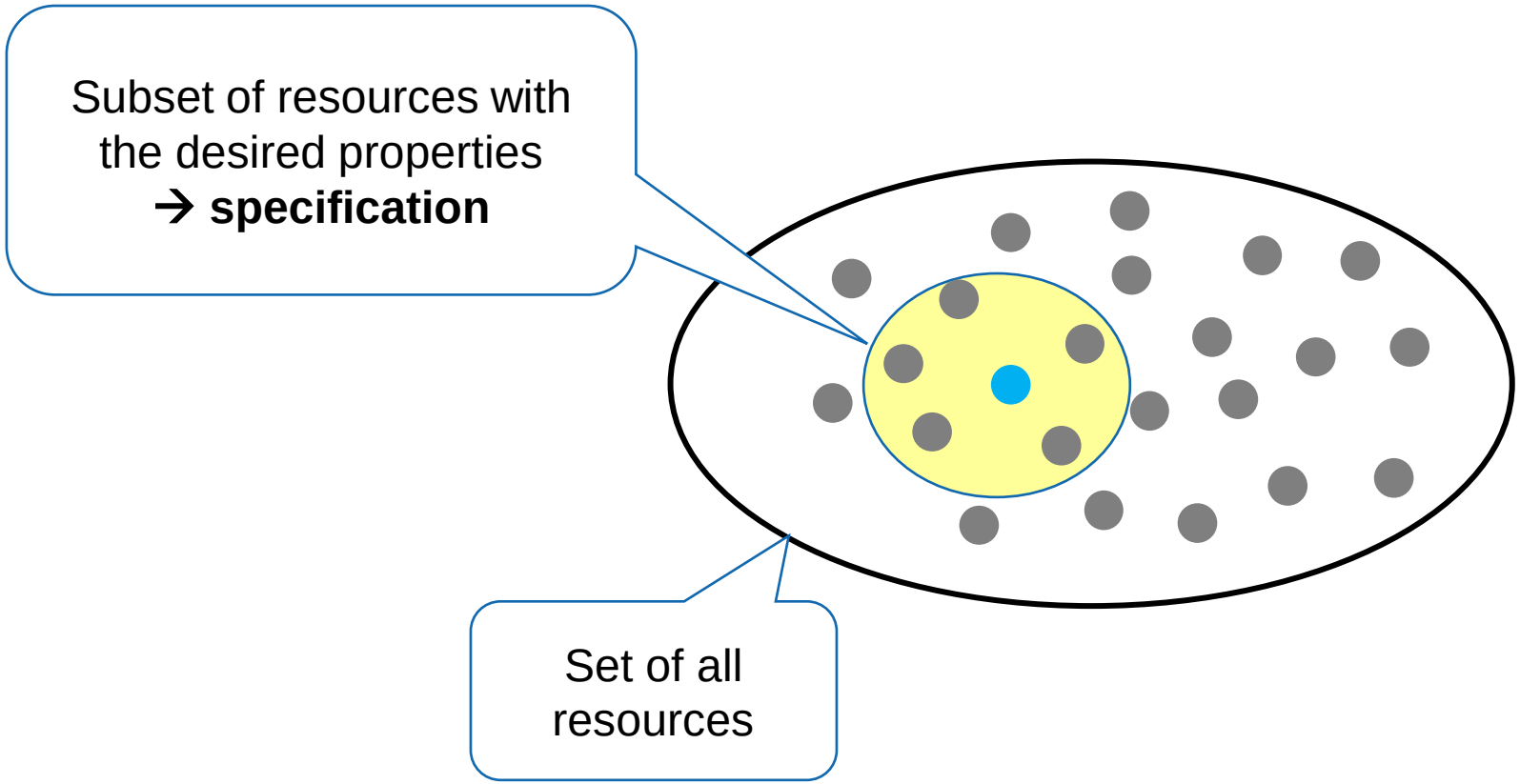
Open question: How would such a notion fit within a composable framework?

Specifications: a fresh take on composable security

Rethinking Composable Security: Specifications



Rethinking Composable Security: Specifications

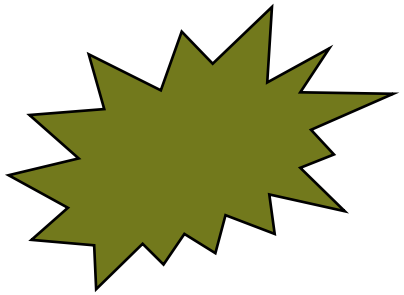


Rethinking Composable Security: Specifications

- General statement: **specification abstraction**
  - Abstract assumed specification by constructed one
  - Easier to understand

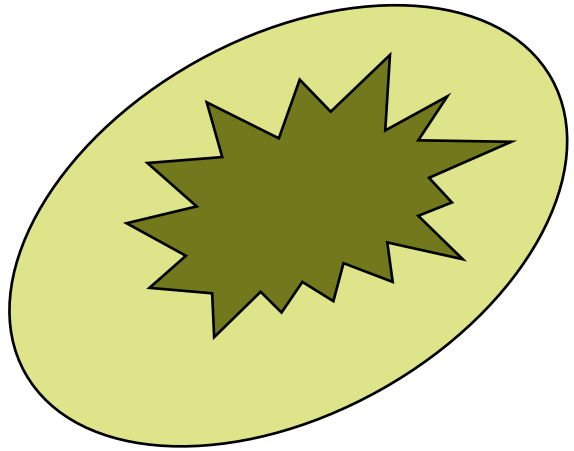
Rethinking Composable Security: Specifications

- General statement: **specification abstraction**
  - Abstract assumed specification by constructed one
  - Easier to understand



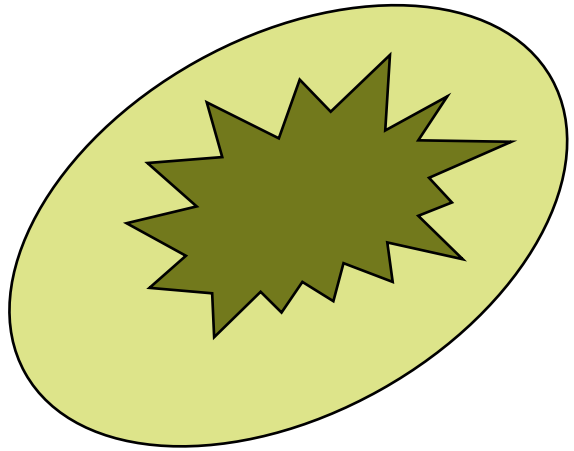
Rethinking Composable Security: Specifications

- General statement: **specification abstraction**
  - Abstract assumed specification by constructed one
  - Easier to understand



Rethinking Composable Security: Specifications

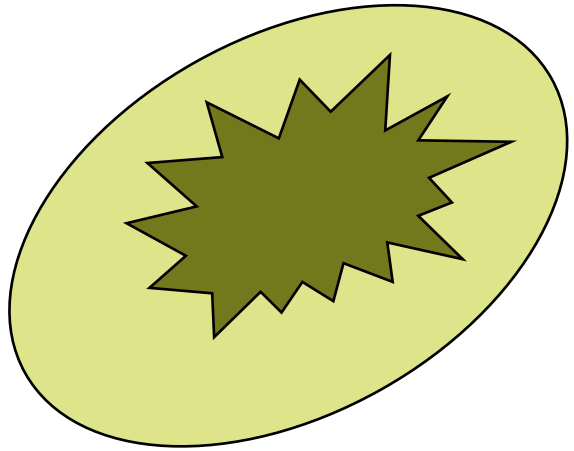
- General statement: **specification abstraction**
  - Abstract assumed specification by constructed one
  - Easier to understand



Introduced in Mau-Ren'16

Rethinking Composable Security: Specifications

- General statement: **specification abstraction**
  - Abstract assumed specification by constructed one
  - Easier to understand



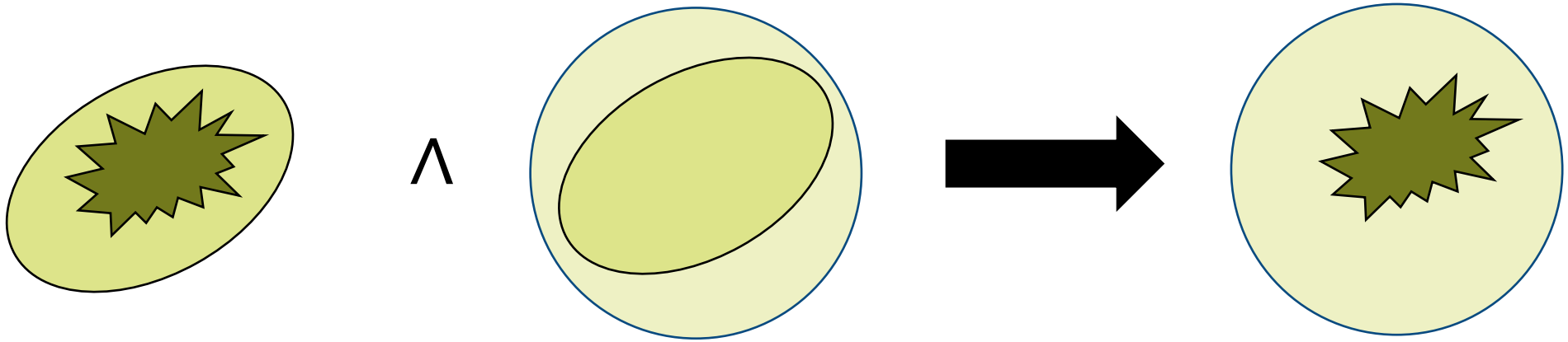
- While traditional composable framework have a single type of statement, specifications give us **flexibility**:
  - Basic properties and compositional guarantees fixed
  - But not the types of specifications!

Rethinking Composable Security: Specifications

- **Advantages:**
  - Absolute statement: no «forgotten» attacks
  - Composition: transitivity of subset relation
  - Intersection

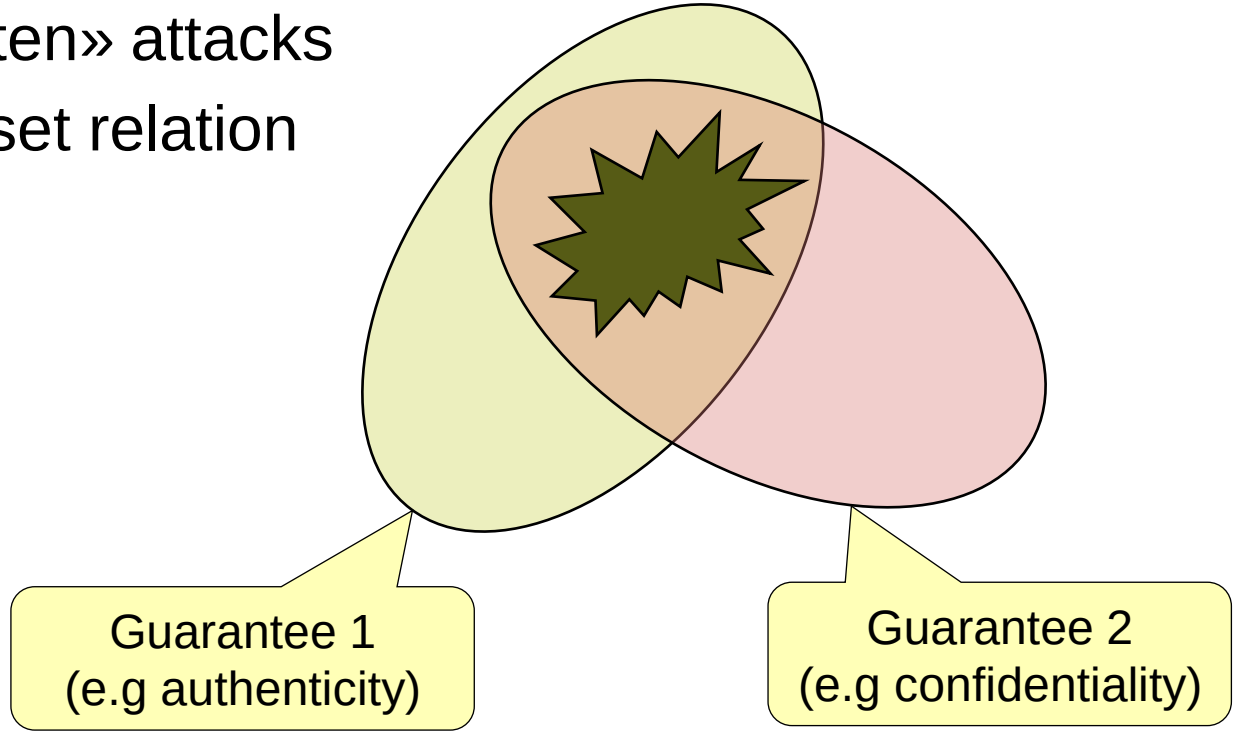
Rethinking Composable Security: Specifications

- **Advantages:**
  - Absolute statement: no «forgotten» attacks
  - Composition: transitivity of subset relation
  - Intersection



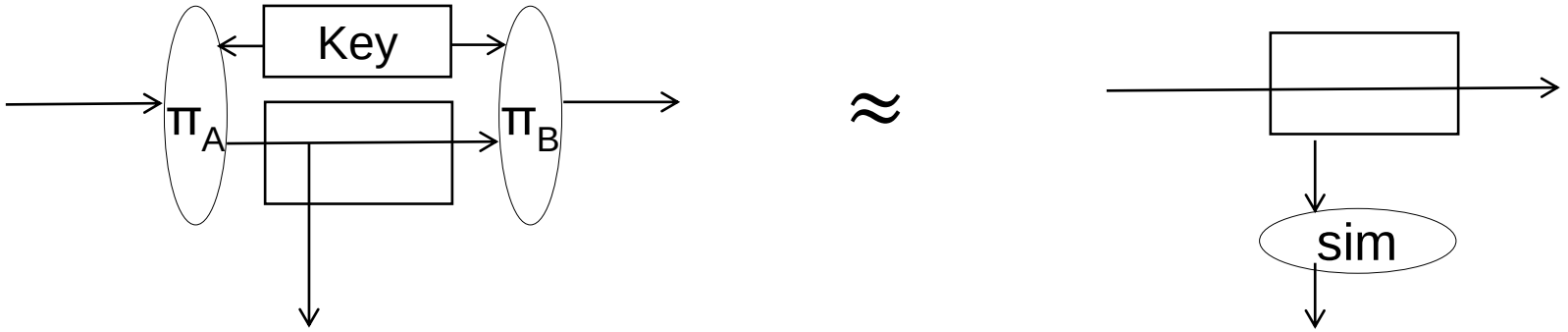
Rethinking Composable Security: Specifications

- **Advantages:**
  - Absolute statement: no «forgotten» attacks
  - Composition: transitivity of subset relation
  - Intersection



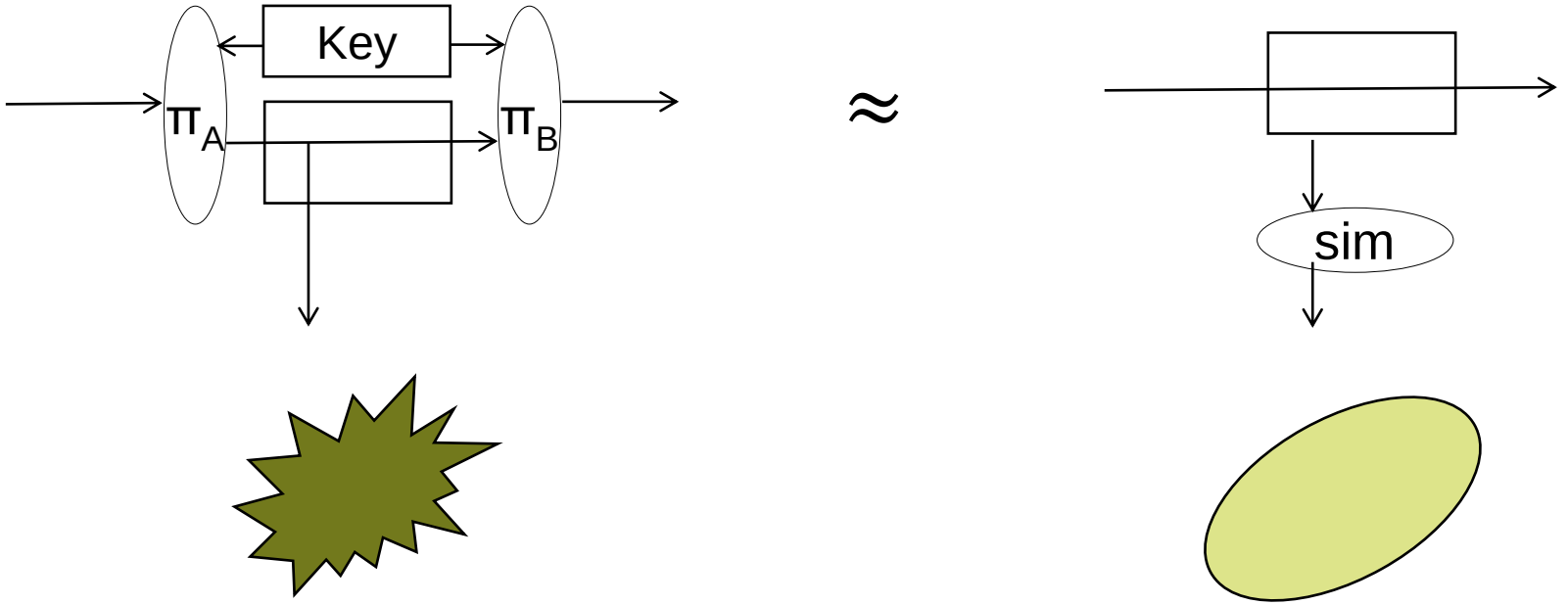
Simulation-based Security

- The standard «simulation-based» notion can be expressed as a special case of specification abstraction:



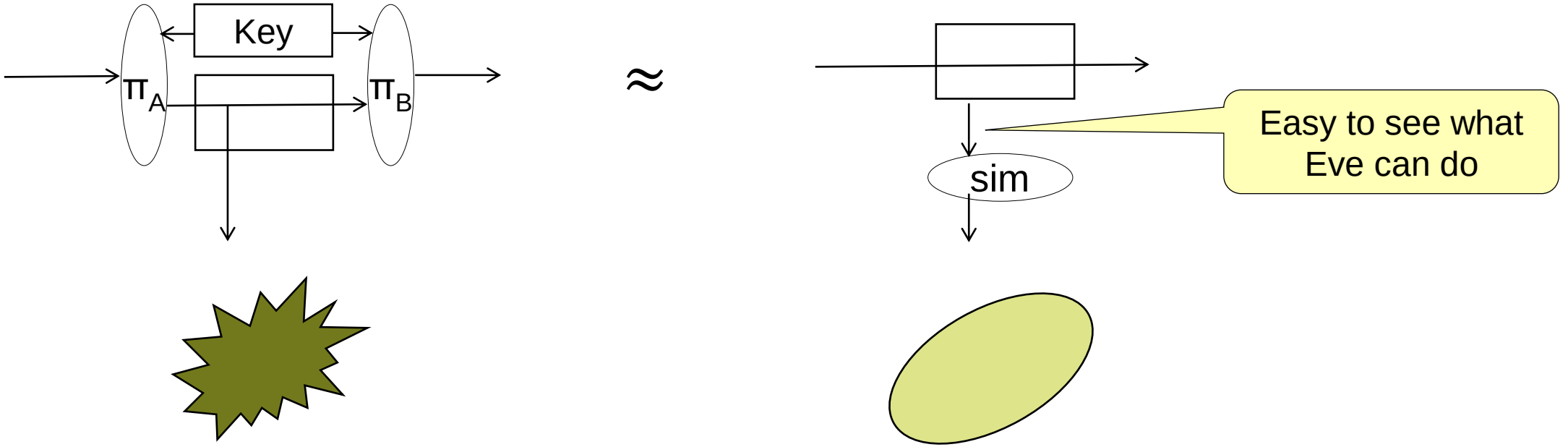
Simulation-based Security

- The standard «simulation-based» notion can be expressed as a special case of specification abstraction:



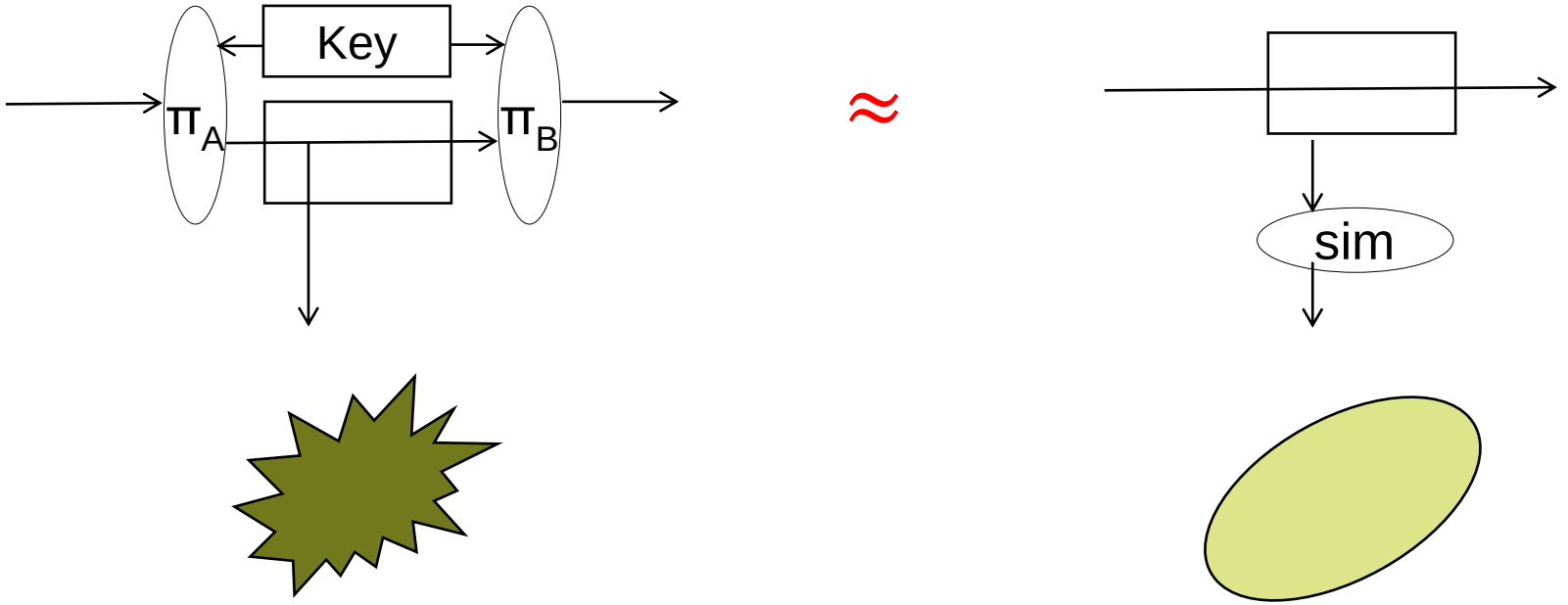
Simulation-based Security

- The standard «simulation-based» notion can be expressed as a special case of specification abstraction:



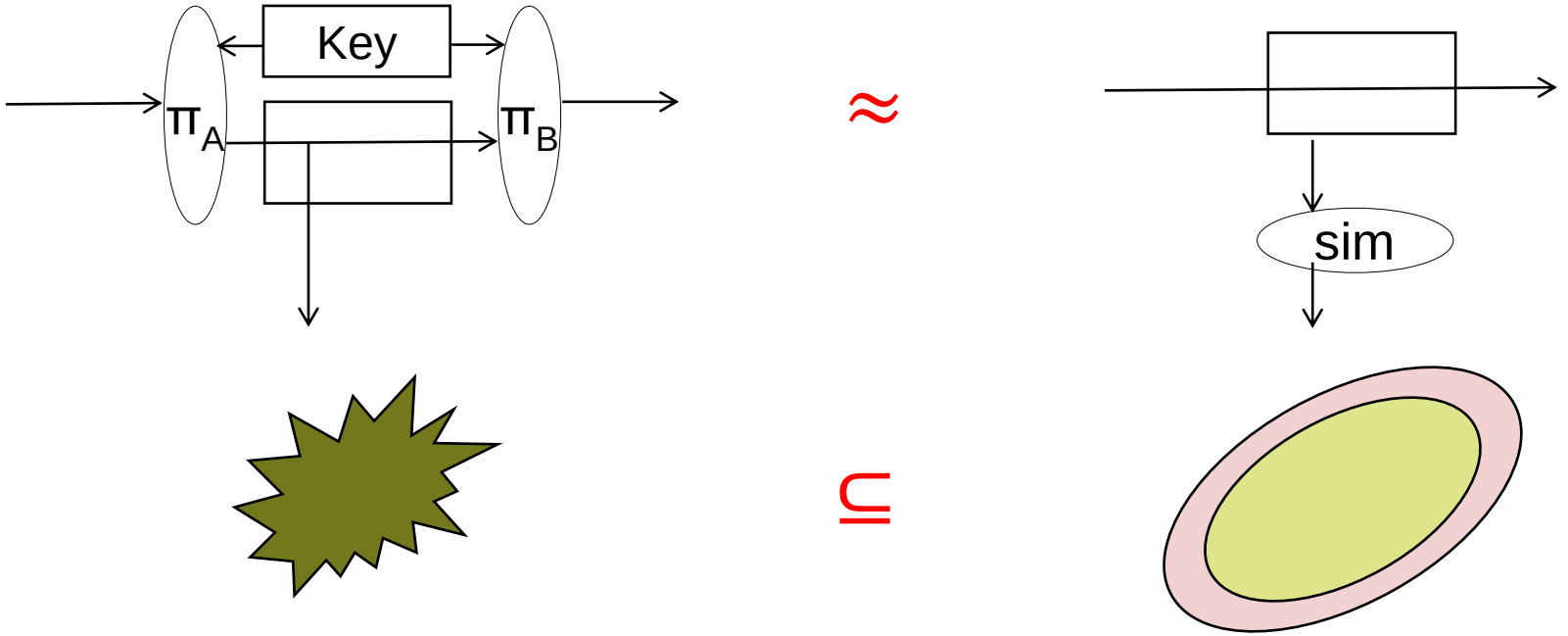
Simulation-based Security

- The standard «simulation-based» notion can be expressed as a special case of specification abstraction:



Simulation-based Security

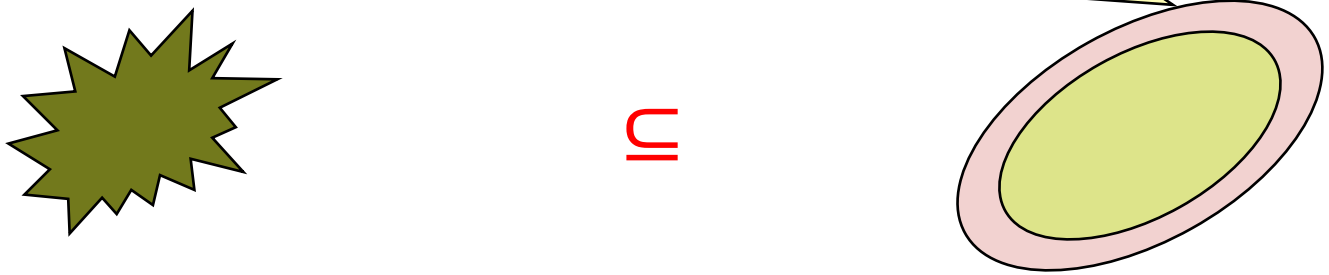
- The standard «simulation-based» notion can be expressed as a special case of specification abstraction:



Simulation-based Security

- The standard «simulation-based» notion can be expressed as a special case of specification abstraction:

ε-relaxation:
the set of all resources that are computationally indistinguishable to one of the original (green) specification.



Simulation-based Security

- The ε-relaxation has two important properties:
  1. Commutes with protocol application

$$\pi(\mathcal{R}^{\epsilon}) \subseteq (\pi\mathcal{R})^{\epsilon}$$

  2. Monotonicity:

$$\mathcal{R} \subseteq \mathcal{S} \implies \mathcal{R}^{\epsilon} \subseteq \mathcal{S}^{\epsilon}$$

Simulation-based Security

- The ε-relaxation has two important properties:

  - The «standard» composition rule can be recovered as a syntactic derivation rule
  - In particular simulator and ε-relaxation can be ignored in further construction step

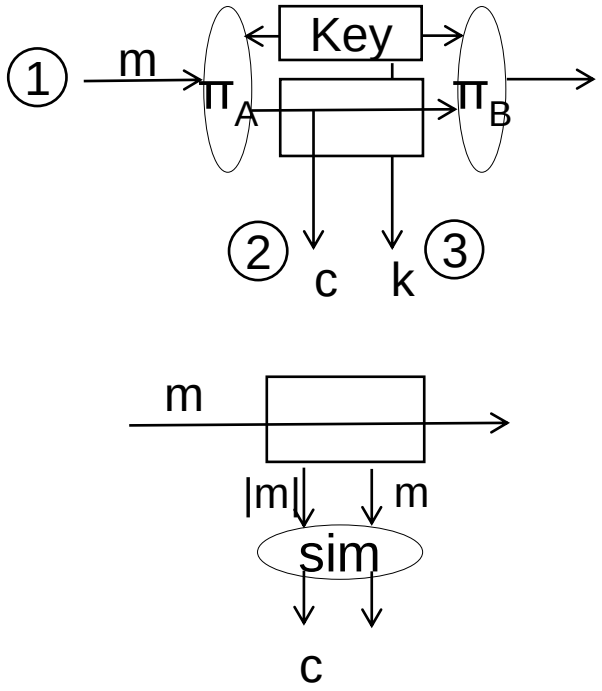
  ➔ **Having structured specifications is crucial for true modularity!**

Interval-wise Relaxations

Interval-wise Guarantees

We use this specification based view this to overcome commitment problem!

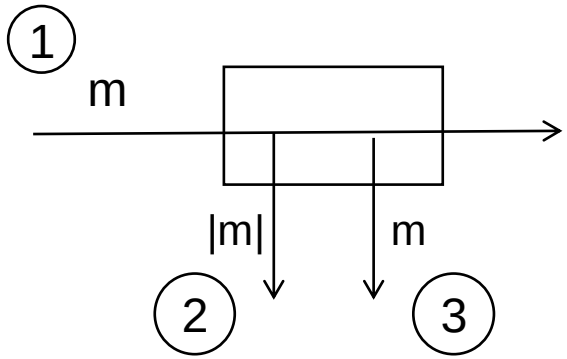
- **Recall:** cannot simulate across exposure of key
- **Solution:** we formalize the guarantees before and after the key exposure as separate specifications:
  1. Confidentiality until the key is exposed
  2. Remaining guarantees afterwards



Interval-wise Guarantees

Formalization of interval-wise guarantees as a specification:

1. Start with unachievable resource (what one might hope for)

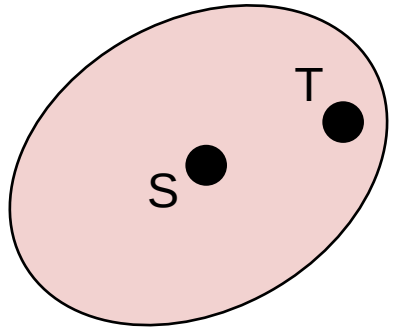


2. Apply suitable relaxations:
   - **Until-rexation**: waives all guarantees after a certain event
   - **From-relaxation**: waives all guarantees before a certain event

Interval-wise Guarantees

Until-relaxation:

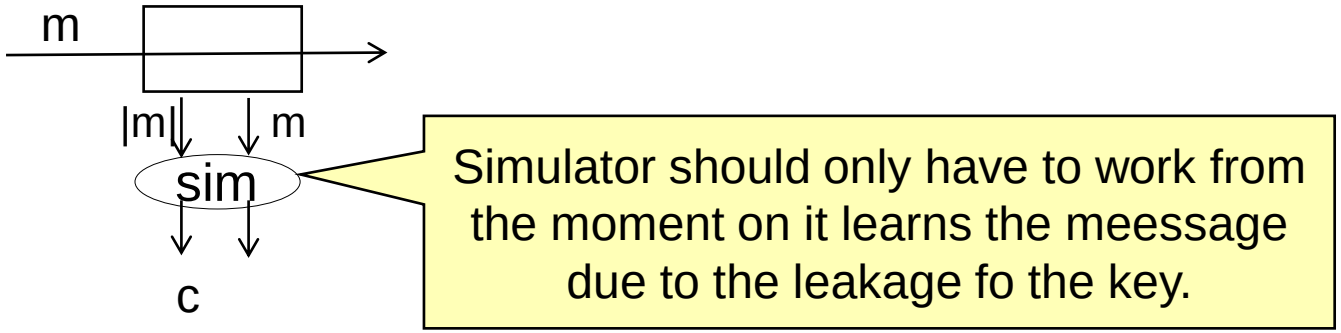
- System S gets relaxed to the set T of systems that behave identically until the event happens.
- Formalized by considering the projection of the systems that no longer replies from the event on.



Interval-wise Guarantees

From-relaxation:

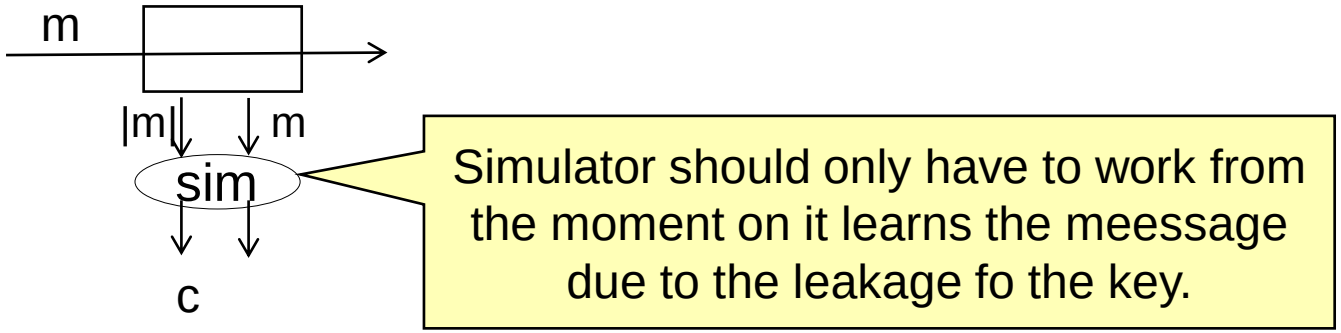
- How to formalize that the interaction only starts at a certain event?



Interval-wise Guarantees

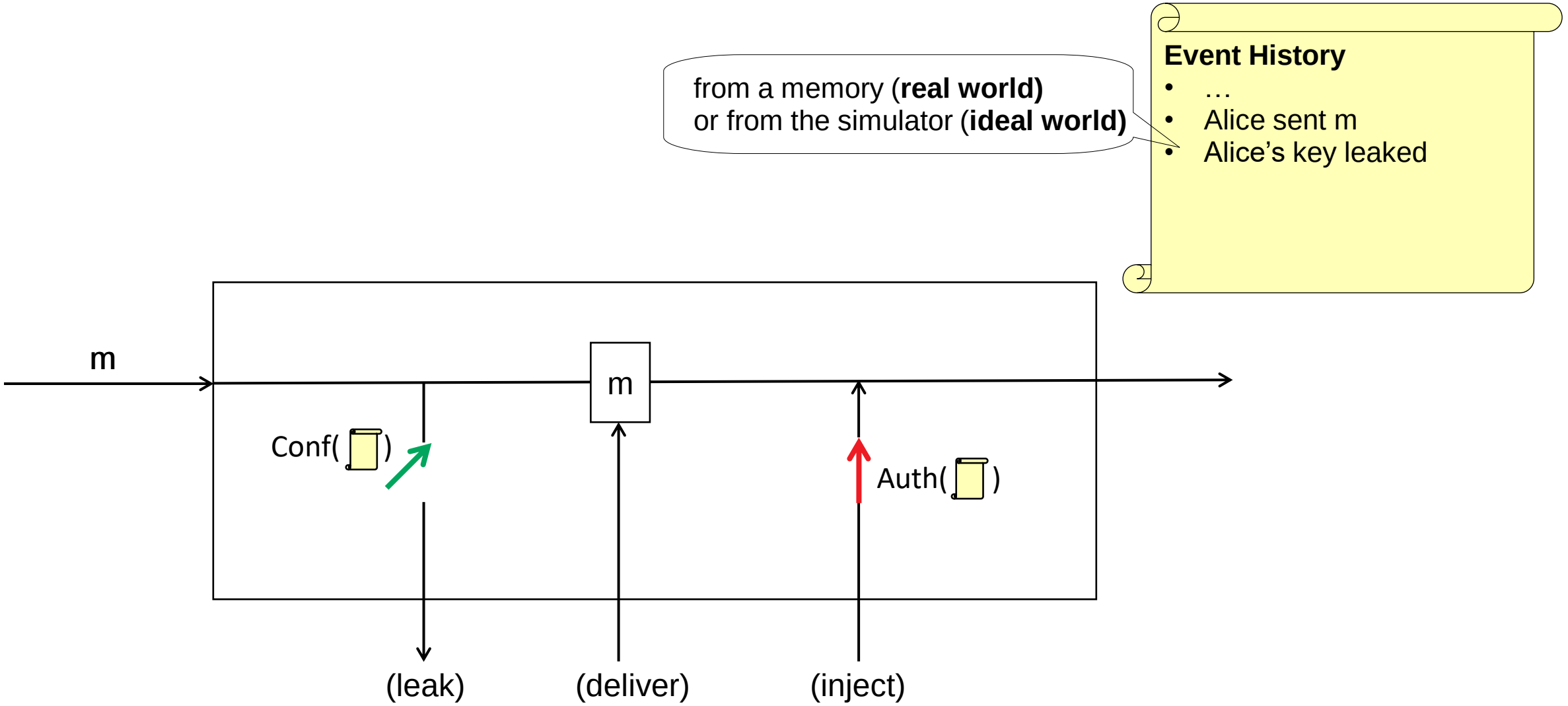
From-relaxation:

- How to formalize that the interaction only starts at a certain event?



- → Solution: only consider so-called *external events* from CC with events [J-Mau-Mul'19b]

Constructive Cryptography with Events



Interval-wise Guarantees

Putting it all together:

- The guarantees for **each interval** are formalized as a specification.
- Each such specification might involve a simulator, but **not necessarily the same** one!
- The interval-wise specifications are built around **relaxations** of the same (overly idealized) resource.
- We show how those relaxations interaction with the ε-relaxation and protocol attachment

  → **Syntactical composition rules**

Examples

We considered two additional examples:

- Identity-based encryption
  - [Hof-Mat-Mau'15] : Simulation-based secure IBE is impossible in the standard model
  - Using interval-wise guarantees we introduce a composable notion that is equivalent to the standard IND-ID-CPA notion.

Examples

We considered two additional examples:

- Identity-based encryption
  - [Hof-Mat-Mau'15] : Simulation-based secure IBE is impossible in the standard model
  - Using interval-wise guarantees we introduce a composable notion that is equivalent to the standard IND-ID-CPA notion.

- Coin-tossing over the phone without setup (via commitments)

Conclusions

1. Specifications allow for a more flexible approach towards composable security.

2. Considering structured specifications: syntactic composition rules.

3. Interval-wise specifications: avoid several impossibility results due to the simulator-commitment problem.

ETH zürich

Thank you for your attention!

Mau-Ren'16 – TCC 2016b

Ueli Maurer and Renato Renner

From indifferentiability to constructive cryptography (and back)

J-Mau-Mul'19b – TCC 2019

Daniel Jost, Ueli Maurer, and Marta Mularczyk
A Unified and Composable Take on Ratcheting

Hof-Mat-Mau'15 – Asiacrypt 2015

Dennis Hofheinz, Christian Matt, and Ueli Maurer
Idealizing Identity-Based Encryption

ETH zürich