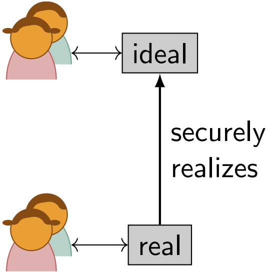
Formalizing Constructive Cryptography using CryptHOL

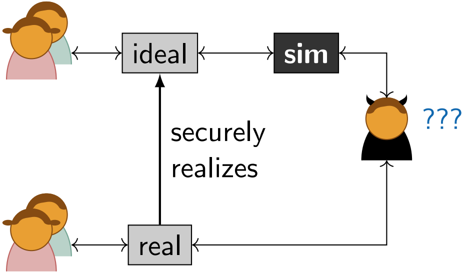
Andreas Lochbihler S. Reza Sefidgar David A. Basin Ueli Maurer

Digital Asset ETH zürich

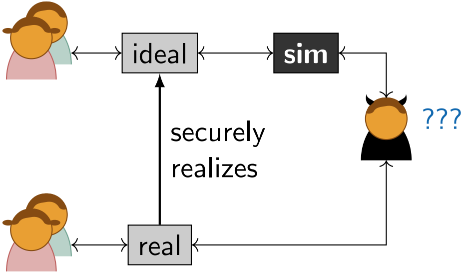
Simulation-based Cryptography



Simulation-based Cryptography



Simulation-based Cryptography



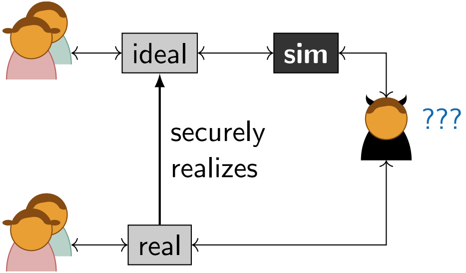
compositionality

Universal Composability

BPW

Constructive Cryptography

Simulation-based Cryptography



compositionality

Universal Composability

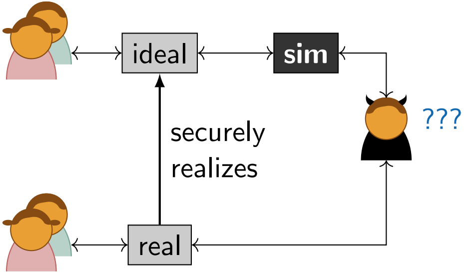
BPW

Constructive Cryptography

Computer-aided Cryptography

Mechanic checks for cryptographic proofs to overcome the crisis of rigour

Simulation-based Cryptography



compositionality

Universal Composability

BPW

Constructive Cryptography

Computer-aided Cryptography

Mechanic checks for cryptographic proofs to overcome the crisis of rigour

CertiCrypt

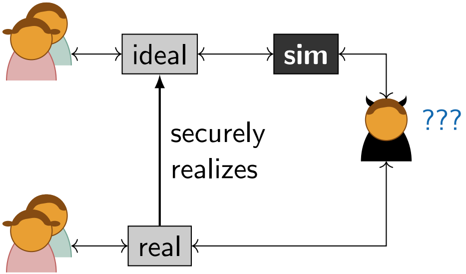
CryptoVerif

EasyCrypt

FCF

CryptHOL

Simulation-based Cryptography



compositionality

Universal Composability

BPW

Constructive Cryptography

Computer-aided Cryptography

Mechanic checks for cryptographic proofs to overcome the crisis of rigour

CertiCrypt

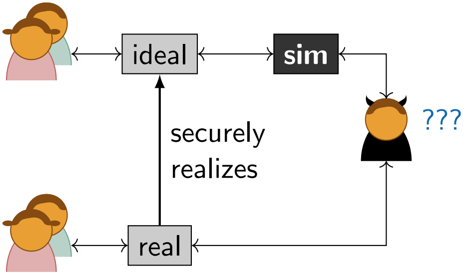
CryptoVerif

EasyCrypt

FCF

CryptHOL

Simulation-based Cryptography



compositionality

Universal Composability

BPW

Constructive Cryptography

Computer-aided Cryptography

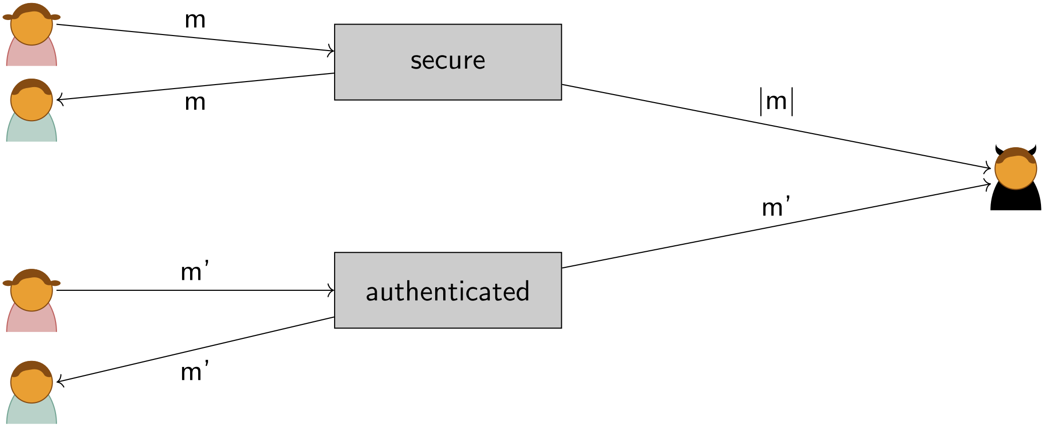
Mechanic checks for cryptographic proofs to overcome the crisis of rigour

In this talk:

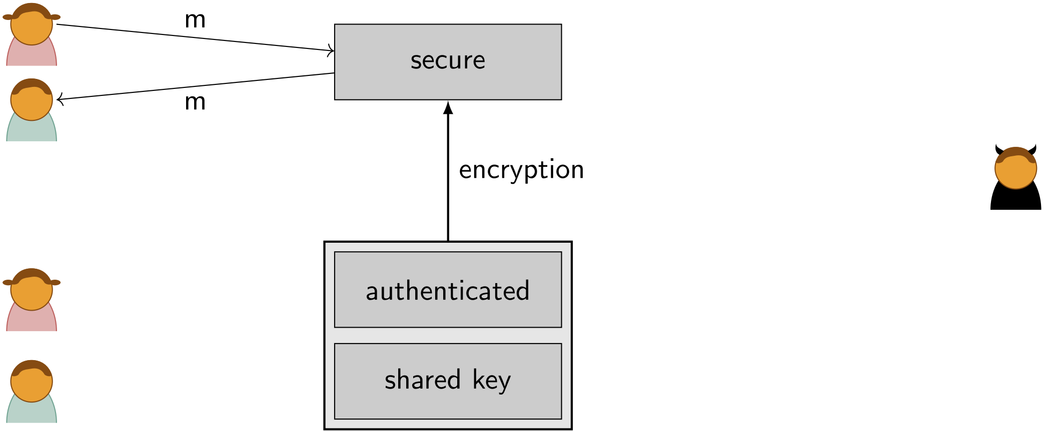
- CC formalization in Isabelle/HOL (information-theoretic security)
- proof of compositionality
- application to a case study (insecure channel $\rightsquigarrow$ secure channel)

CryptHOL

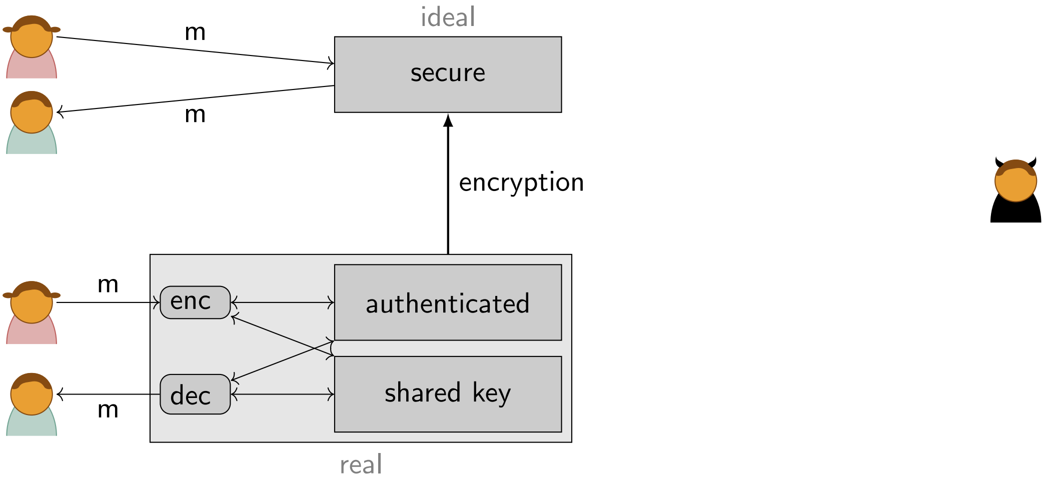
Channels in Constructive Cryptography



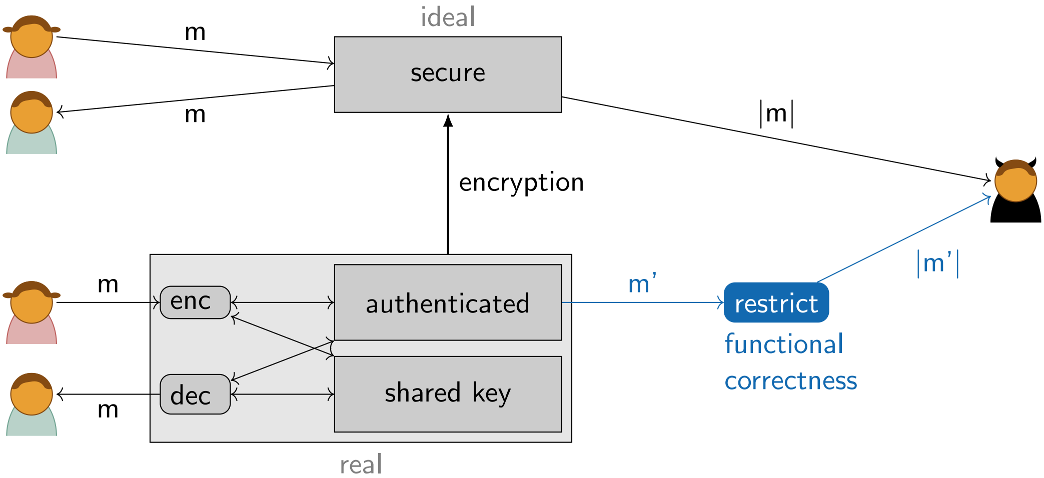
Channels in Constructive Cryptography



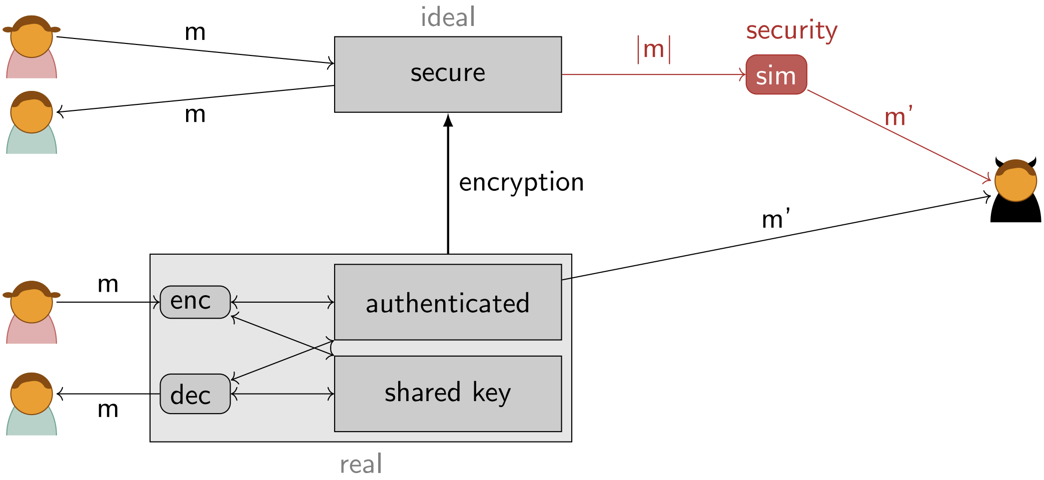
Channels in Constructive Cryptography



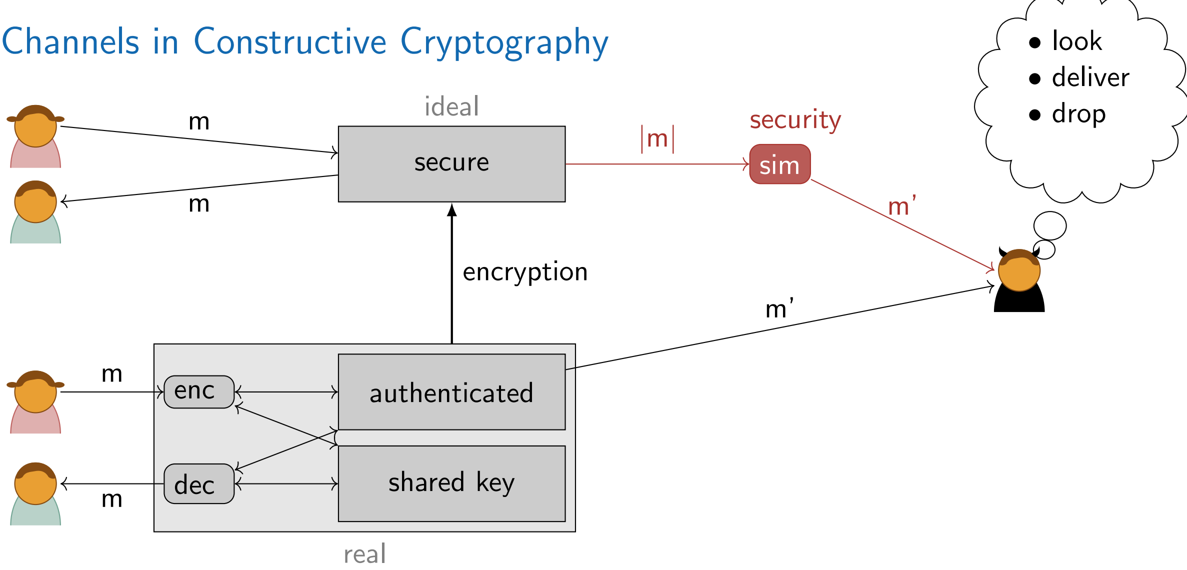
Channels in Constructive Cryptography



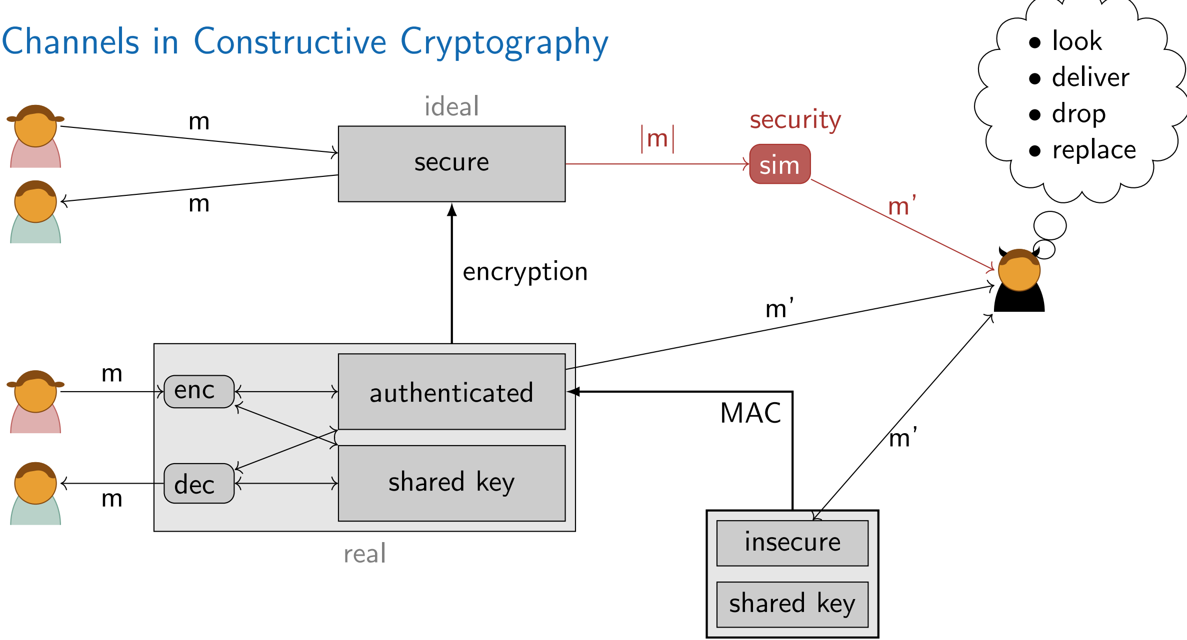
Channels in Constructive Cryptography



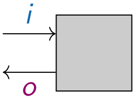
Channels in Constructive Cryptography



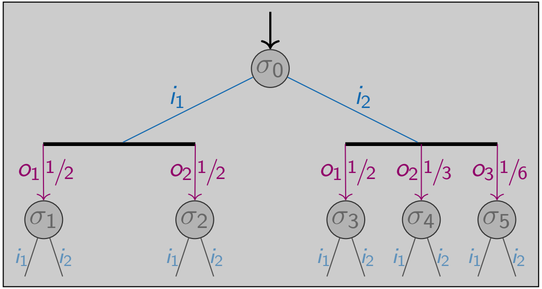
Channels in Constructive Cryptography



Formalizing Resources



⇝

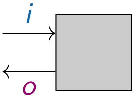


1. Probabilistic transition system (d, σ_0)

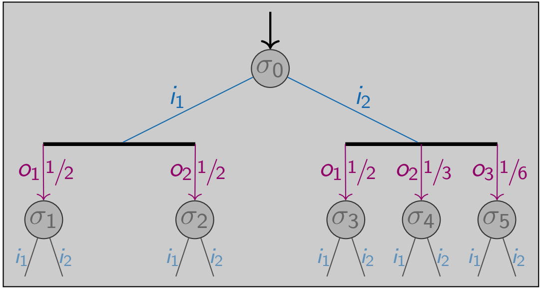
$$
\begin{aligned}
d &: \Sigma \to I \to \mathbb{D}(O \times \Sigma) \\
\sigma_0 &: \Sigma
\end{aligned}
$$

(= CryptHOL oracle)

Formalizing Resources



⇝



1. Probabilistic transition system (d, σ_0)

$$
\begin{aligned}
d &: \Sigma \to I \to \mathbb{D}(O \times \Sigma) \\
\sigma_0 &: \Sigma
\end{aligned}
$$

(= CryptHOL oracle)

---

2. Abstract over the concrete state

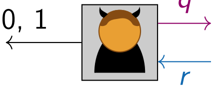
$$\exists \Sigma.\ (\Sigma \to I \to \mathbb{D}(O \times \Sigma)) \times \Sigma$$

codatatype $\mathbb{R}(I, O) =$
$\quad$ $\mathit{Resource}\ (I \to \mathbb{D}(O \times \mathbb{R}(I, O)))$

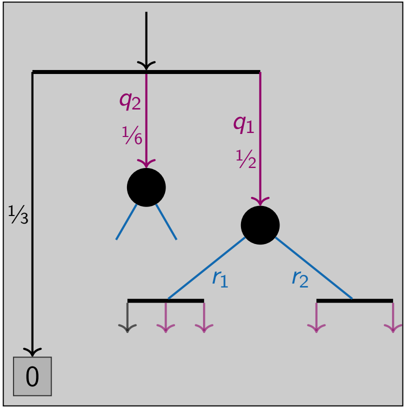
Benefits

- Identifies bisimilar resources
- Can exploit **corecursive structure** (unwinding) in definitions and proofs

Formalizing Distinguishers (≈ CryptHOL Adversary)



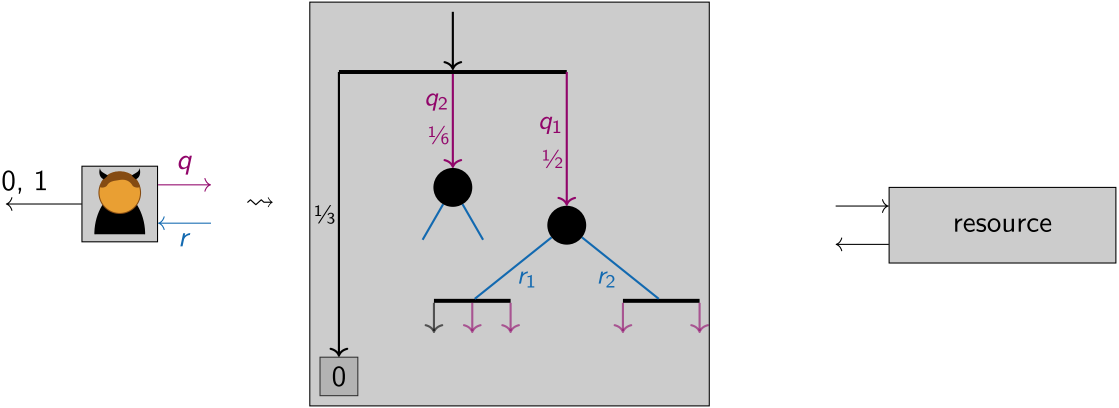
⇝



CryptHOL: Generative probabilistic value (GPV) + probabilistic termination

$$\texttt{codatatype}\ \mathbb{G}(A, Q, R) = Gpv\ (\mathbb{D}(A + (Q \times (R \to \mathbb{G}(A, Q, R)))))$$

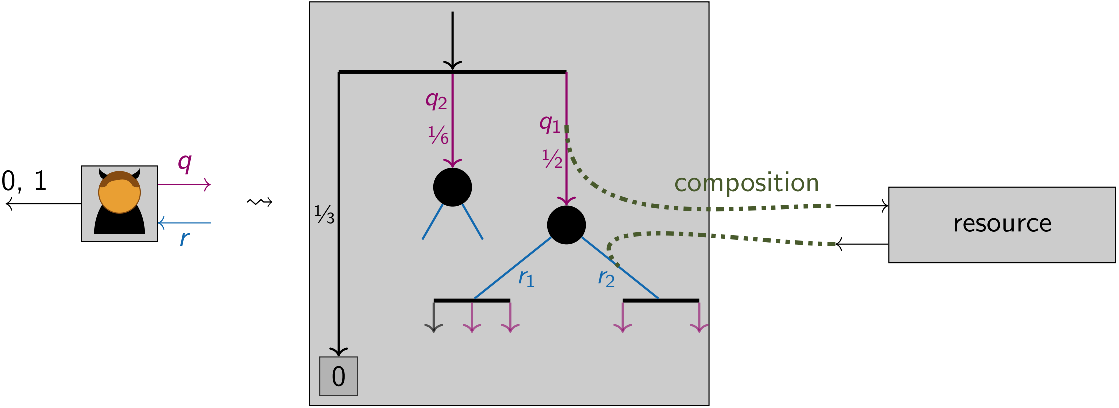
Formalizing Distinguishers (≈ CryptHOL Adversary)



CryptHOL: Generative probabilistic value (GPV) + probabilistic termination

$$\texttt{codatatype}\ \mathbb{G}(A, Q, R) = \mathit{Gpv}\ (\mathbb{D}(A + (Q \times (R \to \mathbb{G}(A, Q, R)))))$$

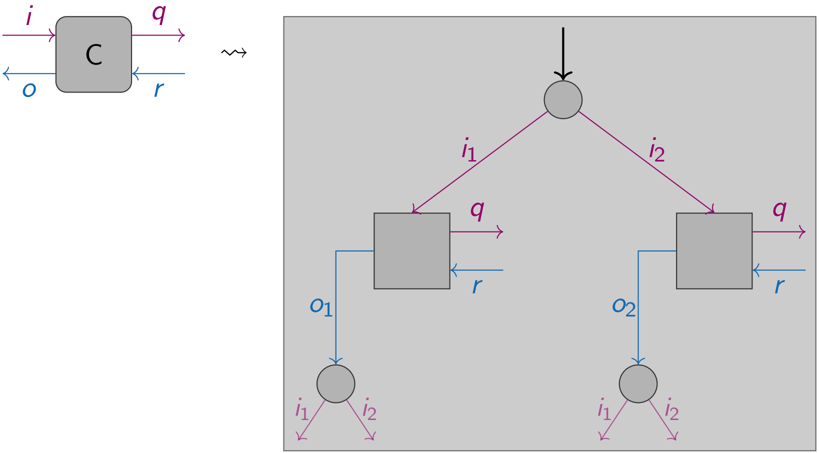
Formalizing Distinguishers (≈ CryptHOL Adversary)



CryptHOL: Generative probabilistic value (GPV) + probabilistic termination

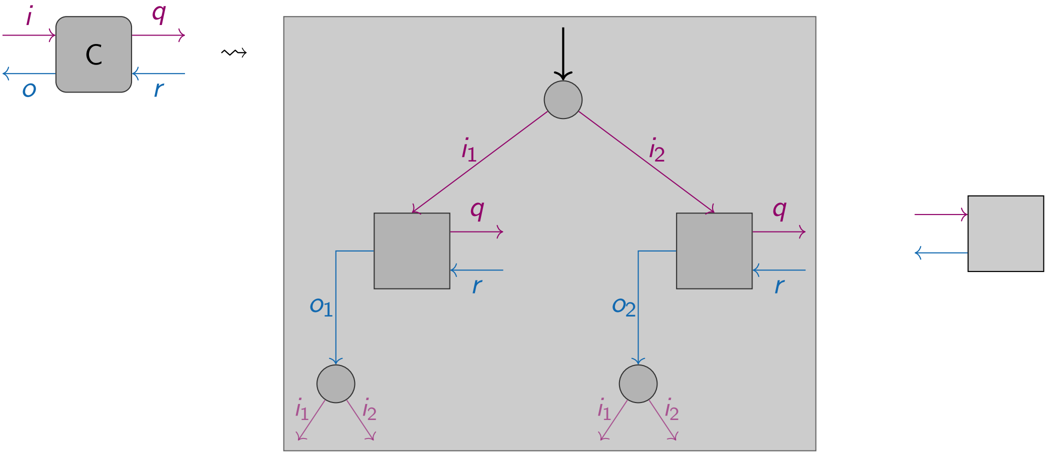
$$\texttt{codatatype}\ \mathbb{G}(A, Q, R) = \mathit{Gpv}\ (\mathbb{D}(A + (Q \times (R \to \mathbb{G}(A, Q, R)))))$$

Formalizing Converters



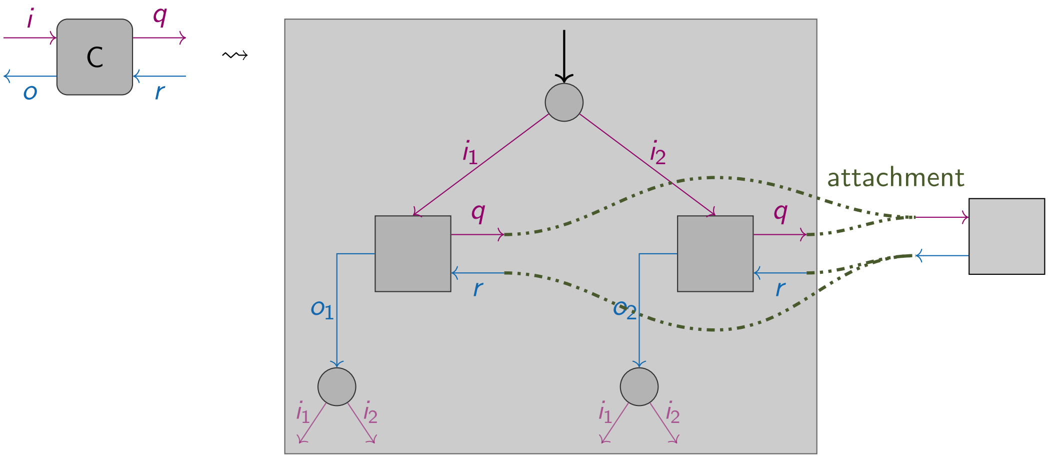
$$\texttt{codatatype}\ \mathbb{C}(I, O, Q, R) = \mathit{Converter}\ (I \to \mathbb{G}(O \times \mathbb{C}(I, O, Q, R), Q, R))$$

Formalizing Converters



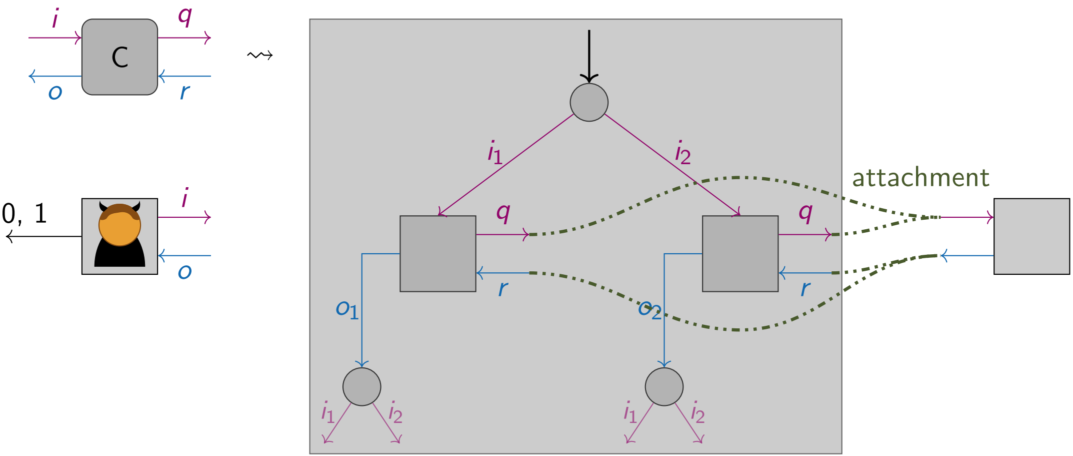
$$\texttt{codatatype}\ \mathbb{C}(I, O, Q, R) = \mathit{Converter}\ (I \to \mathbb{G}(O \times \mathbb{C}(I, O, Q, R), Q, R))$$

Formalizing Converters



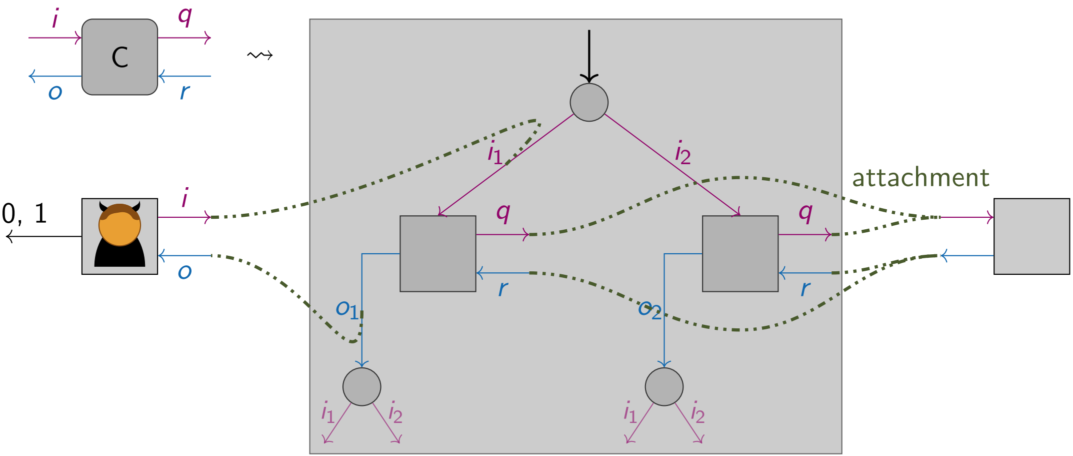
$$\texttt{codatatype } \mathbb{C}(I, O, Q, R) = \textit{Converter } (I \to \mathbb{G}(O \times \mathbb{C}(I, O, Q, R), Q, R))$$

Formalizing Converters



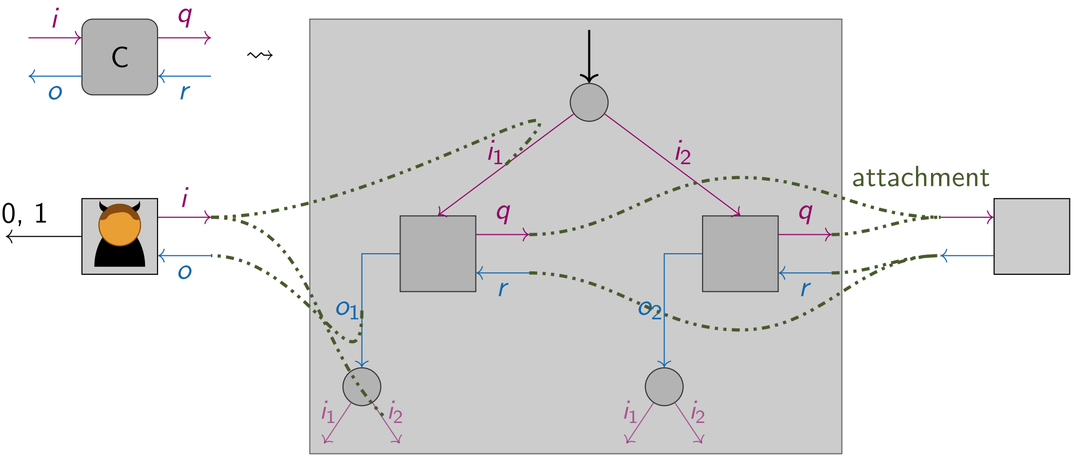
$$\texttt{codatatype}\ \mathbb{C}(I, O, Q, R) = \mathit{Converter}\ (I \to \mathbb{G}(O \times \mathbb{C}(I, O, Q, R), Q, R))$$

Formalizing Converters



$$\texttt{codatatype}\ \mathbb{C}(I, O, Q, R) = \mathit{Converter}\ (I \to \mathbb{G}(O \times \mathbb{C}(I, O, Q, R), Q, R))$$

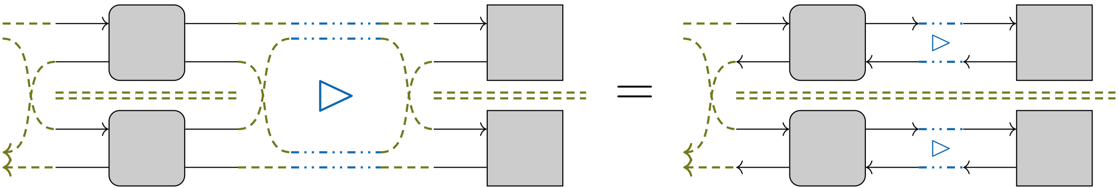
Formalizing Converters



$$\texttt{codatatype}\ \mathbb{C}(I, O, Q, R) = \mathit{Converter}\ (I \to \mathbb{G}(O \times \mathbb{C}(I, O, Q, R), Q, R))$$

Algebraic Reasoning

```
lemma attach_parallel2:
  "(C1 |= C2) ▷ (R1 ∥ R2) = (C1 ▷ R1) ∥ (C2 ▷ R2)"
```



Algebraic Reasoning Becomes Simpler

Abstraction over state simplifies reasoning about composition

```
lemma attach_compose:
  "(C1 ⊙ C2) ▷ R = C1 ▷ (C2 ▷ R)"
```

Algebraic Reasoning Becomes Simpler

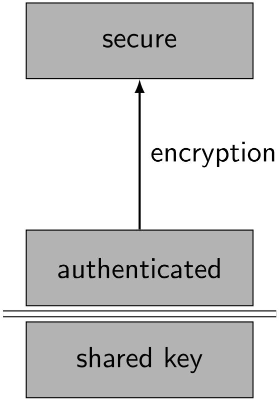
Abstraction over state simplifies reasoning about composition

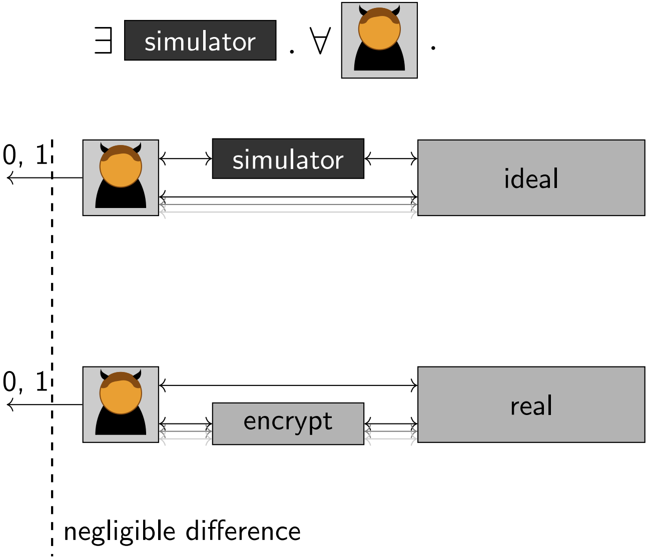
```
lemma attach_compose:
  "(C1 ⊙ C2) ▷ R = C1 ▷ (C2 ▷ R)"
```

In CryptHOL:

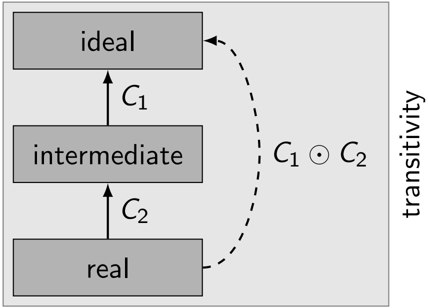
```
lemma exec_gpv_inline:
  "exec_gpv R (inline C2 C1 s') s =
   map_spmf (λ(x, s', s). ((x, s'), s)) (exec_gpv
     (λ(s', s) y. map_spmf (λ((x, s'), s). (x, s', s))
       (exec_gpv R (C2 s' y) s))
     C1 (s', s))"
```

Formalizing Secure Realization (asymptotic version)

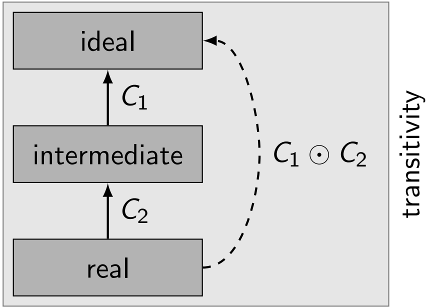


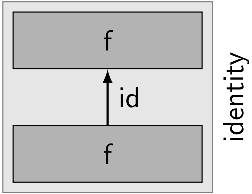


Formalized Composition Theorems

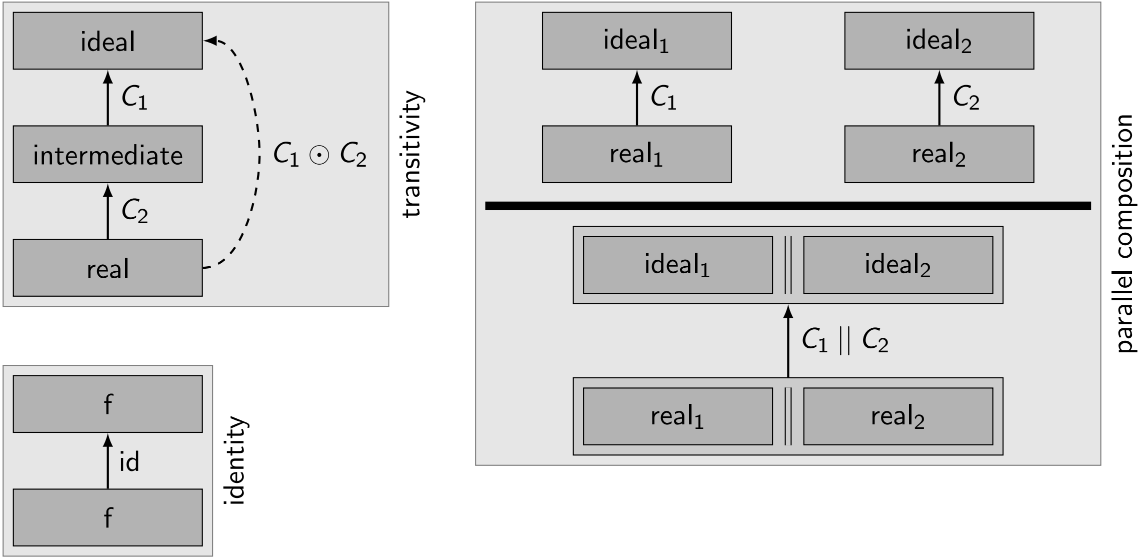


Formalized Composition Theorems





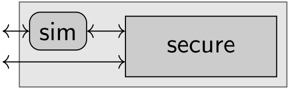
Formalized Composition Theorems



Example: One-time-pad Encryption over a Single-use Channel

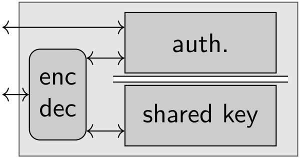
Interfaces

Resource	Users	Adversary
secure channel	submit / poll	**length**, deliver, drop
authenticated ch.	submit / poll	**look**, deliver, drop
shared key	get	—



Encrypt:

1. get key
2. XOR key with message
3. submit



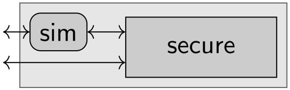
Decrypt:

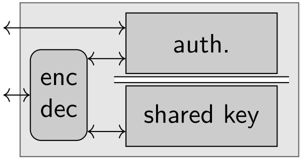
1. get key
2. poll message
3. XOR key with message

Example: One-time-pad Encryption over a Single-use Channel

Interfaces

Resource	Users	Adversary
secure channel	submit / poll	**length**, deliver, drop
authenticated ch.	submit / poll	**look**, deliver, drop
shared key	get	—





Encrypt:

1. get key
2. XOR key with message
3. submit

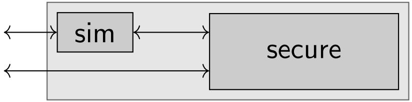
Decrypt:

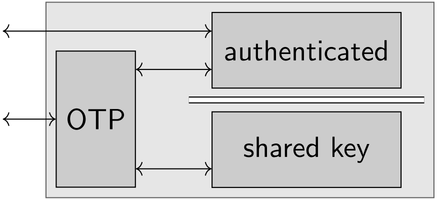
1. get key
2. poll message
3. XOR key with message

Simulator:

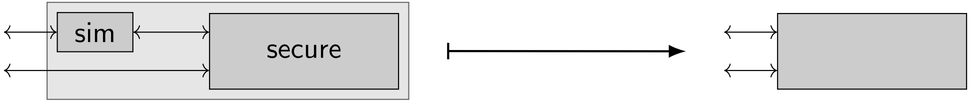
authenticated	$\mapsto$	secure channel
look	$\mapsto$	length + sample bitstring
deliver	$\mapsto$	deliver
drop	$\mapsto$	drop

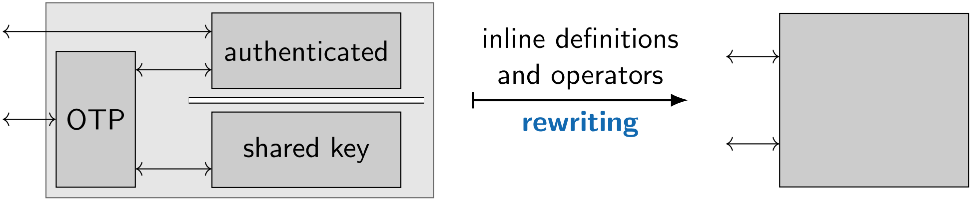
Proof Approach



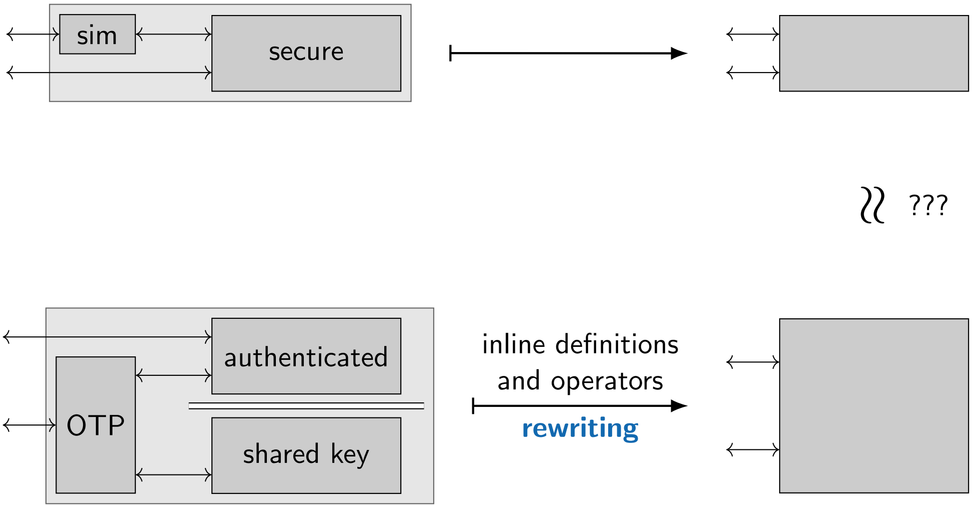


Proof Approach

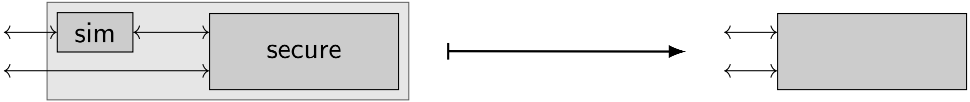




Proof Approach

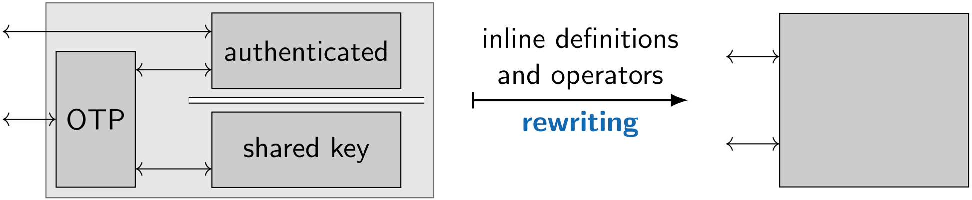


Proof Approach

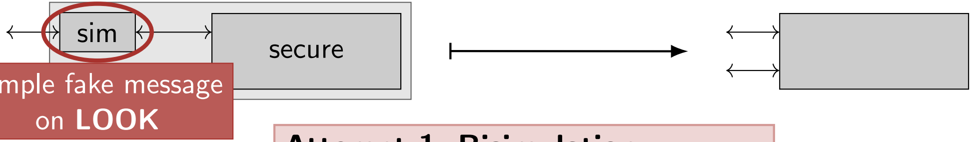


Attempt 1: Bisimulation
relation between states of the resources must be preserved by every interaction
$\rightsquigarrow$ local reasoning

$\wr$???

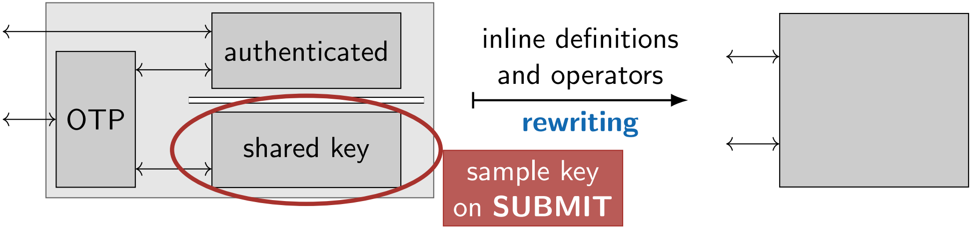


Proof Approach

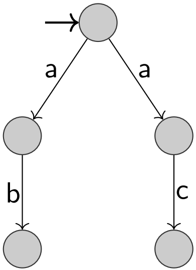


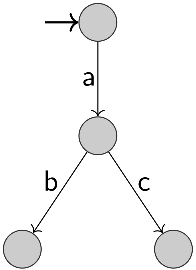
Attempt 1: Bisimulation
relation between states of the resources must be preserved by every interaction
⇝ local reasoning

≀ ???



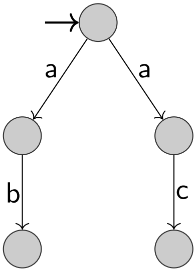
Why Bisimulation is too Strong



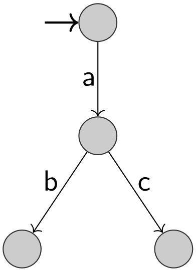


Why Bisimulation is too Strong

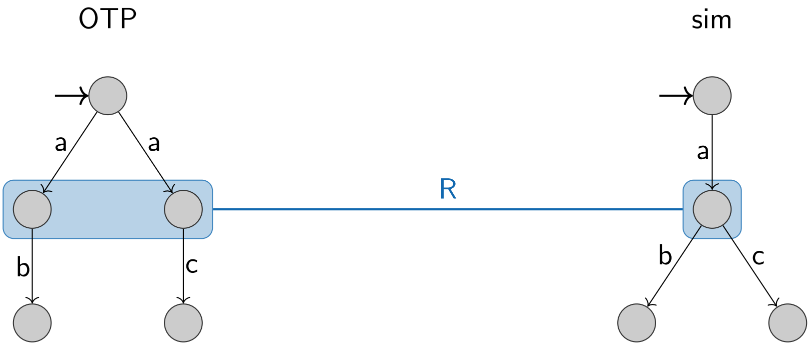
OTP



sim



Why Bisimulation is too Strong



Attempt 2: Trace Equivalence

Random system [Maurer'02]: Family of conditional probability distributions

previous interactions		next input		conditional distribution over next output
$[I \times O]$	$\rightarrow$	I	$\rightarrow$	$\mathbb{D}(O)$

$$(\Sigma \rightarrow I \rightarrow \mathbb{D}(O \times \Sigma)) \times \quad \Sigma$$

Attempt 2: Trace Equivalence

Random system [Maurer'02]: Family of conditional probability distributions

previous interactions		next input		conditional distribution over next output
$[I \times O]$	$\rightarrow$	I	$\rightarrow$	$\mathbb{D}(O)$

$$\Uparrow \textit{trace}$$

$$(\Sigma \rightarrow I \rightarrow \mathbb{D}(O \times \Sigma)) \times \quad \Sigma$$

Attempt 2: Trace Equivalence

Random system [Maurer'02]: Family of conditional probability distributions

previous interactions		next input		conditional distribution over next output
$[I \times O]$	$\rightarrow$	I	$\rightarrow$	$\mathbb{D}(O)$

$\Uparrow$ *trace*

recursive definition

$$(\Sigma \rightarrow I \rightarrow \mathbb{D}(O \times \Sigma)) \times \mathbb{D}(\Sigma)$$

Attempt 2: Trace Equivalence

Random system [Maurer'02]: Family of conditional probability distributions

previous interactions		next input		conditional distribution over next output
$[I \times O]$	$\rightarrow$	I	$\rightarrow$	$\mathbb{D}(O)$

$\Uparrow$ *trace* recursive definition

$$(\Sigma \rightarrow I \rightarrow \mathbb{D}(O \times \Sigma)) \times \mathbb{D}(\Sigma)$$

Characterization theorem:
Two resources are trace equivalent
iff the distinguishing advantage is 0.

Attempt 2: Trace Equivalence

Random system [Maurer'02]: Family of conditional probability distributions

previous interactions		next input		conditional distribution over next output
$[I \times O]$	$\rightarrow$	I	$\rightarrow$	$\mathbb{D}(O)$

$\Uparrow$ *trace* recursive definition

$$(\Sigma \rightarrow I \rightarrow \mathbb{D}(O \times \Sigma)) \times \mathbb{D}(\Sigma)$$

Characterization theorem:
Two resources are trace equivalent
iff the distinguishing advantage is 0.

Sound and complete **unwinding proof rule**
Local, simulation-like proof principle
for trace equivalence

Attempt 2: Trace Equivalence

Random system [Maurer'02]: Family of conditional probability distributions

previous interactions		next input		conditional distribution over next output
$[I \times O]$	$\rightarrow$	I	$\rightarrow$	$\mathbb{D}(O)$

Suffices to complete the proofs

definition

$$\| $$

$$(\Sigma \rightarrow I \rightarrow \mathbb{D}(O \times \Sigma)) \times \mathbb{D}(\Sigma)$$

Characterization theorem:
Two resources are trace equivalent
iff the distinguishing advantage is 0.

Sound and complete **unwinding proof rule**
Local, simulation-like proof principle
for trace equivalence

Limitations and Comparison

Limitations:

- Information-theoretic security
- Linear interactions (**pull model**)

Limitations and Comparison

Limitations:

- Information-theoretic security
- Linear interactions (**pull model**)

	CryptHOL	**FCF**	**EasyCrypt**
Underlying technology	Isabelle/HOL	Coq	OCaml
Definitional approach	⊕	⊕	⊕
Expressive codatatypes	⊕	0	⊖
Library	⊕	0	growing
Dependent types	⊖	⊕	⊖

Take aways

1. Coalgebraic modelling $\rightsquigarrow$ mechanized algebraic reasoning
2. Trace equivalence is the right equivalence notion
3. Unwinding proof rule for trace equivalence
4. Formalization suitable for abstract (composition) and concrete (OTP, MAC) reasoning

`www.isa-afp.org/entries/Constructive_Cryptography.html`

Take aways

1. Coalgebraic modelling $\leadsto$ mechanized algebraic reasoning
2. Trace equivalence is the right equivalence notion
3. Unwinding proof rule for trace equivalence
4. Formalization suitable for abstract (composition) and concrete (OTP, MAC) reasoning

More in the paper

- Dependent type system
  for resources and converters
- Formalization of wiring

`www.isa-afp.org/entries/Constructive_Cryptography.html`

Take aways

1. Coalgebraic modelling $\rightsquigarrow$ mechanized algebraic reasoning
2. Trace equivalence is the right equivalence notion
3. Unwinding proof rule for trace equivalence
4. Formalization suitable for abstract (composition) and concrete (OTP, MAC) reasoning

More in the paper

- Dependent type system for resources and converters
- Formalization of wiring

Future work

- Further applications
- Computational security

`www.isa-afp.org/entries/Constructive_Cryptography.html`