Formalizing Constructive Cryptography using CryptHOL

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

Digital Asset

ETH Zürich
Simulation-based Cryptography

- ideal
- securely realizes
- real

In this talk:
- CC formalization in Isabelle/HOL (information-theoretic security)
- proof of compositionality
- application to a case study (insecure channel $\Rightarrow$ secure channel)
Simulation-based Cryptography

- Ideal
- Sim
- Securely realizes
- Real

In this talk:
- CC formalization in Isabelle/HOL (information-theoretic security)
- Proof of compositionality
- Application to a case study (insecure channel → secure channel)
Simulation-based Cryptography

Compositionality

- Universal Composability
- BPW
- Constructive Cryptography

In this talk:
- CC formalization in Isabelle/HOL (information-theoretic security)
- Proof of compositionality
- Application to a case study (insecure channel $\Rightarrow$ secure channel)
Simulation-based Cryptography

Computer-aided Cryptography

Mechanic checks for cryptographic proofs to overcome the crisis of rigour

compositionality

Universal Composability

BPW

Constructive Cryptography
Simulation-based Cryptography

Computer-aided Cryptography

Mechanic checks for cryptographic proofs to overcome the crisis of rigour

CertiCrypt

CryptoVerif

EasyCrypt

FCF

CryptHOL

Universal Composability

BPW

Constructive Cryptography

In this talk:

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

Computer-aided Cryptography

Mechanic checks for cryptographic proofs to overcome the crisis of rigour

CertiCrypt
CryptoVerif
EasyCrypt
FCF

Compositionality

Universal Composability
BPW
Constructive Cryptography
CryptHOL

In this talk:
• CC formalization in Isabelle/HOL (information-theoretic security)
• proof of compositionality
• application to a case study (insecure channel \(\Rightarrow\) secure channel)
Simulation-based Cryptography

Ideal \xrightarrow{\text{sim}} \text{real} \xleftarrow{\text{securely realizes}} \text{ideal}

compositionality

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 $\leadsto$ secure channel)
Channels in Constructive Cryptography

- Secure
- Authenticated

- m
- m
- m
- m'
- m'
- m'
- m'
- m'
Channels in Constructive Cryptography

- Secure
- Encryption
- Authenticated
- Shared key

- Look
- Deliver
- Drop
- Replace

- Insecure
- Shared key
- MAC
Channels in Constructive Cryptography

- ideal
- secure
- encryption
- real
- authenticated
- shared key
- enc
- dec
- m
- m'
- m
- m'
Channels in Constructive Cryptography

ideal

secure

encryption

restrict

functional correctness

authenticated

shared key

enc

dec

real

A. Lochbihler (Digital Asset, ETH Zurich)
Channels in Constructive Cryptography

A. Lochbihler (Digital Asset, ETH Zurich)
Form. Constructive Cryptog. using CryptHOL
Channels in Constructive Cryptography

- ideal
- real
- secure
- encryption
- authenticated
- shared key
- m
- m'
- look
- deliver
- drop
- sim
- security
- enc
- dec
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
- m
- m'
Channels in Constructive Cryptography

- look
- deliver
- drop
- replace

- ideal
- real
- secure
- enc
- dec
- authenticated
- shared key
- encryption
- security
- sim
- m
- m'
- m'
- MAC
- insecure
- shared key
Formalizing Resources

1. Probabilistic transition system \((d, \sigma_0)\)

\[
d : \Sigma \rightarrow I \rightarrow \mathbb{D}(O \times \Sigma)
\]

\[
\sigma_0 : \Sigma
\]

\(=\) CryptHOL oracle
Formalizing Resources

1. Probabilistic transition system \((d, \sigma_0)\)

\[ d : \Sigma \to I \to \mathbb{D}(O \times \Sigma) \]
\[ \sigma_0 : \Sigma \]

(= CryptHOL oracle)

2. Abstract over the concrete state

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

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

Benefits

- Identifies bisimilar resources
- Can exploit corecursive structure (unwinding) in definitions and proofs
Formalizing Distinguishers ($\approx$ CryptHOL Adversary)

CryptHOL: Generative probabilistic value (GPV) + probabilistic termination

$$\text{codatatype } \mathcal{G}(A, Q, R) = \text{Gpv} \ (\mathbb{D}(A + (Q \times (R \to \mathcal{G}(A, Q, R))))$$
Formalizing Distinguishers ($\approx$ CryptHOL Adversary)

CryptHOL: Generative probabilistic value (GPV) + probabilistic termination

\[
\text{codatatype } G(A, Q, R) = \text{Gpv}(\mathbb{D}(A + (Q \times (R \rightarrow G(A, Q, R))))))
\]
Formalizing Distinguishers (∼ CryptHOL Adversary)

CryptHOL: Generative probabilistic value (GPV) + probabilistic termination

\[ \text{codatatype } \mathbb{G}(A, Q, R) = \text{Gpv} \left( \mathbb{D}(A + (Q \times (R \to \mathbb{G}(A, Q, R)))) \right) \]
Formalizing Converters

\[ \text{codatatype } \mathbb{C}(I, O, Q, R) = \text{Converter } (I \rightarrow \mathbb{G}(O \times \mathbb{C}(I, O, Q, R), Q, R)) \]
Formalizing Converters

\[
\text{codatatype } C(I, O, Q, R) = \text{Converter } (I \rightarrow G(O \times C(I, O, Q, R), Q, R))
\]
Formalizing Converters

\[
\text{codatatype } \mathbb{C}(I, O, Q, R) = \text{Converter } (I \rightarrow \mathbb{G}(O \times \mathbb{C}(I, O, Q, R), Q, R))
\]
Formalizing Converters

\[ C \xrightarrow{\sim} 0, 1 \]

\[ \text{codatatype } C(I, O, Q, R) = \text{Converter } (I \rightarrow \mathbb{G}(O \times C(I, O, Q, R), Q, R)) \]
Formalizing Converters

codatatype $\mathbb{C}(I, O, Q, R) = \text{Converter} \ (I \rightarrow \mathcal{G}(O \times \mathbb{C}(I, O, Q, R), Q, R))$
Formalizing Converters

codatatype \( \mathbb{C}(I, O, Q, R) = \text{Converter} (I \to \mathbb{G}(O \times \mathbb{C}(I, O, Q, R), Q, R)) \)
**lemma** attach\_parallel2:

"(C1 \|= C2) \triangleright (R1 \parallel R2) = (C1 \triangleright R1) \parallel (C2 \triangleright 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

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

In CryptHOL:

```text
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)

∃ simulator . ∀ .

secure

encryption

authenticated

shared key

negligible difference

0, 1

0, 1

encrypt

ideal

real
Formalized Composition Theorems

\[ C_1 \circ C_2 \]

transitivity

ideal \[\uparrow C_1\]
intermediate \[\uparrow C_2\]
real
Formalized Composition Theorems

\[ C_1 \circ C_2 \]

- **ideal**
- **intermediate**
- **real**

**transitivity**

- \( C_1 \circ C_2 \)

**identity**

- \( f \)
- \( \text{id} \)
- \( f \)
Formalized Composition Theorems

### Transitivity

\[ C_1 \circ C_2 \]

### Parallel Composition

\[ C_1 \parallel C_2 \]
Example: One-time-pad Encryption over a Single-use Channel

**Interfaces**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Users</th>
<th>Adversary</th>
</tr>
</thead>
<tbody>
<tr>
<td>secure channel</td>
<td>submit / poll</td>
<td>length, deliver, drop</td>
</tr>
<tr>
<td>authenticated ch.</td>
<td>submit / poll</td>
<td>look, deliver, drop</td>
</tr>
<tr>
<td>shared key</td>
<td>get</td>
<td></td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Users</th>
<th>Adversary</th>
</tr>
</thead>
<tbody>
<tr>
<td>secure channel</td>
<td>submit / poll</td>
<td>length, deliver, drop</td>
</tr>
<tr>
<td>authenticated ch.</td>
<td>submit / poll</td>
<td>look, deliver, drop</td>
</tr>
<tr>
<td>shared key</td>
<td>get</td>
<td></td>
</tr>
</tbody>
</table>

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

A. Lochbihler (Digital Asset, ETH Zurich)
Proof Approach

- sim
- secure

- authenticated
- shared key

- OTP

Attempt 1: Bisimulation relation between states of the resources must be preserved by every interaction \( \Rightarrow \) local reasoning

Sample fake message on LOOK sample key on SUBMIT
Proof Approach

- **sim** → **secure**

- **OTP** → **authenticated** → **shared key**

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

-Inline definitions and operators rewriting
Proof Approach

1. **Attempt 1: Bisimulation**
   - A bisimulation relation between states of the resources must be preserved by every interaction.
   - \( \Rightarrow \) local reasoning

2. **Sample fake message**
   - Sample fake message on LOOK
   - Sample key on SUBMIT

3. **Inline definitions and operators**
   - Inline definitions and operators
   - **Rewriting**
Proof Approach

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

\[ \rightsquigarrow \]
local reasoning

inline definitions and operators

rewriting
Proof Approach

**Attempt 1: Bisimulation**

relation between states of the resources must be preserved by every interaction

\[\rightsquigarrow\]  

local reasoning

### inline definitions and operators

rewriting

sample fake message on **LOOK**

sample key on **SUBMIT**
Why Bisimulation is too Strong
Why Bisimulation is too Strong

OTP

sim
Why Bisimulation is too Strong

OTP

sim

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

Attempt 2: Trace Equivalence

previous interactions \rightarrow conditional distribution over next output

\[ [I \times O] \rightarrow I \rightarrow D(O) \]

\[(\sum \rightarrow I \rightarrow D(O \times \sum)) \times \Sigma\]
Attempt 2: Trace Equivalence

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

\[
[l \times O] \rightarrow l \rightarrow \mathbb{D}(O)
\]

\[
(\Sigma \rightarrow l \rightarrow \mathbb{D}(O \times \Sigma)) \times \Sigma
\]
Attempt 2: Trace Equivalence

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

\[
[I \times O] \rightarrow I \rightarrow \mathbb{D}(O) \uparrow \text{trace} \downarrow \text{recursive definition} \\
(\Sigma \rightarrow I \rightarrow \mathbb{D}(O \times \Sigma)) \times \mathbb{D}(\Sigma)
\]
Attempt 2: Trace Equivalence

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

previous interactions \[I \times O\] → \[I\] → \[D(O)\]

conditional distribution

next input

over next output

trace recursive definition

\((\Sigma \rightarrow I \rightarrow D(O \times \Sigma)) \times D(\Sigma)\)

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

\[
\begin{align*}
\text{previous interactions} & \quad \text{conditional distribution} \\
[I \times O] & \quad \rightarrow & \quad I & \quad \rightarrow & \quad \mathbb{D}(O) \\
\uparrow & \quad \text{trace} & \quad \text{recursive definition} \\
(\Sigma \rightarrow I \rightarrow \mathbb{D}(O \times \Sigma)) \times \mathbb{D}(\Sigma)
\end{align*}
\]

**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

\[
[I \times O] \rightarrow I \rightarrow \mathbb{D}(O)
\]

previous interactions \hspace{2cm} conditional distribution over next output
\hspace{2cm} next input

Suffices to complete the proofs

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

**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)

<table>
<thead>
<tr>
<th></th>
<th>CryptHOL</th>
<th>FCF</th>
<th>EasyCrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underlying technology</td>
<td>Isabelle/HOL</td>
<td>Coq</td>
<td>OCaml</td>
</tr>
<tr>
<td>Definitional approach</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Expressive codatatypes</td>
<td>+</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Library</td>
<td>+</td>
<td>0</td>
<td>growing</td>
</tr>
<tr>
<td>Dependent types</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

A. Lochbihler (Digital Asset, ETH Zurich) Form. Constructive Cryptog. using CryptHOL
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

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 $\leadsto$ 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

Future work
- Further applications
- Computational security

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