Bounded-Deducibility Security: Reasoning About Information-Flow Security in a Fine-Grained Manner

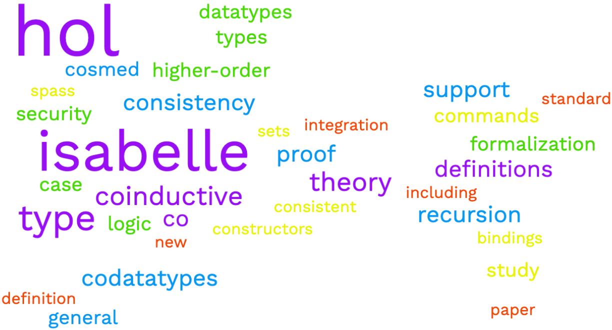
Andrei Popescu

Department of Computer Science
University of Sheffield

ITP 2021

29 June, 2021

ITP & I



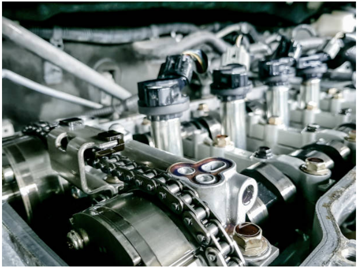
hol
datatypes
types
cosmed
higher-order
spass
support
standard
security
consistency
commands
sets
integration
isabelle
formalization
proof
definitions
theory
case
including
coinductive
consistent
recursion
type
logic
co
constructors
bindings
new
study
codatatypes
definition
paper
general

Isabelle & I

Isabelle view from Urbana, IL

Isabelle & I

Isabelle view from Urbana, IL



Isabelle view from Munich

In This Talk

Verification of information flow security of multi-user, web-based systems

Fine-grained coverage of the (dis)allowed flows

General framework for specifying and verifying such systems

Experience with deploying one such system “in the wild”

In This Talk

❶ Introduction

❷ Bounded-Deducibility (BD) Security

❸ CoCon's Verification

❹ CoCon in the Wild

❺ Other Applications and Developments of BD Security

Contributors to the Work Presented Here

Thomas
Bauereiss

Peter
Lammich

Contributors to the Work Presented Here

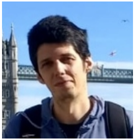
Thomas Bauereiss

Peter Lammich

Sergey Grebenshchikov

Ping Hou

Sudeep Kanav



Armando Pesenti Gritti

Franco Raimondi

Conference Management Systems Going Wrong

EasyChair, the most popular conference management system

HotCRP, the second most popular one

How can they go wrong?

Conference Management Systems Going Wrong

EasyChair, the most popular conference management system

It is our pleasure to inform you that your paper has been accepted to the IEEE Symposium of Security and Privacy (Oakland) 2012. Out of 307 submitted papers, we accepted 40 papers.

HotCRP, the second most popular one

How can they go wrong?

Conference Management Systems Going Wrong

EasyChair, the most popular conference management system

It is our pleasure to inform you that your paper has been accepted to the IEEE Symposium of Security and Privacy (Oakland) 2012. Out of 307 submitted papers, we accepted 40 papers.

We are sorry to inform you that your paper was not accepted for this year's conference. We apologize for an earlier "acceptance" notification. It was due to a system error.

HotCRP, the second most popular one

How can they go wrong?

Conference Management Systems Going Wrong

EasyChair, the most popular conference management system

It is our pleasure to inform you that your paper has been accepted to the IEEE Symposium of Security and Privacy (Oakland) 2012. Out of 307 submitted papers, we accepted 40 papers.

We are sorry to inform you that your paper was not accepted for this year's conference. We apologize for an earlier "acceptance" notification. It was due to a system error.

HotCRP, the second most popular one

How can they go wrong?

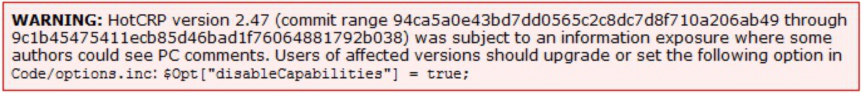
Conference Management Systems Going Wrong

EasyChair, the most popular conference management system

It is our pleasure to inform you that your paper has been accepted to the IEEE Symposium of Security and Privacy (Oakland) 2012. Out of 307 submitted papers, we accepted 40 papers.

We are sorry to inform you that your paper was not accepted for this year's conference. We apologize for an earlier "acceptance" notification. It was due to a system error.

HotCRP, the second most popular one



WARNING: HotCRP version 2.47 (commit range 94ca5a0e43bd7dd0565c2c8dc7d8f710a206ab49 through 9c1b45475411ecb85d46bad1f76064881792b038) was subject to an information exposure where some authors could see PC comments. Users of affected versions should upgrade or set the following option in `Code/options.inc: $Opt["disableCapabilities"] = true;`

How can they go wrong?

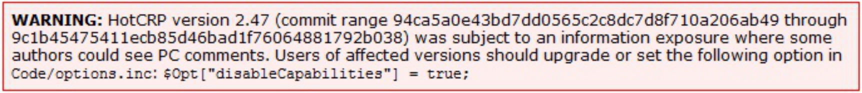
Conference Management Systems Going Wrong

EasyChair, the most popular conference management system

It is our pleasure to inform you that your paper has been accepted to the IEEE Symposium of Security and Privacy (Oakland) 2012. Out of 307 submitted papers, we accepted 40 papers.

We are sorry to inform you that your paper was not accepted for this year's conference. We apologize for an earlier "acceptance" notification. It was due to a system error.

HotCRP, the second most popular one

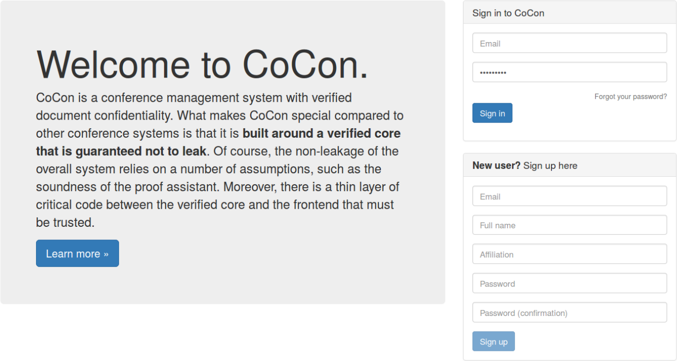


WARNING: HotCRP version 2.47 (commit range 94ca5a0e43bd7dd0565c2c8dc7d8f710a206ab49 through 9c1b45475411ecb85d46bad1f76064881792b038) was subject to an information exposure where some authors could see PC comments. Users of affected versions should upgrade or set the following option in `Code/options.inc`: `$Opt["disableCapabilities"] = true;`

How can they go wrong? Confidentiality and integrity violations

CoCon

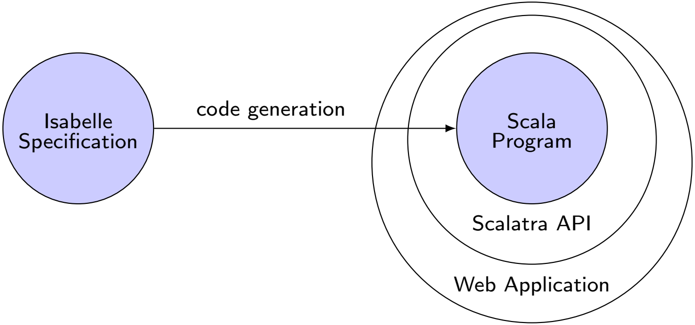
CoCon = Feature-rich conference management system (similar to EasyChair and HotCRP)



Welcome to CoCon.

CoCon is a conference management system with verified document confidentiality. What makes CoCon special compared to other conference systems is that it is **built around a verified core that is guaranteed not to leak**. Of course, the non-leakage of the overall system relies on a number of assumptions, such as the soundness of the proof assistant. Moreover, there is a thin layer of critical code between the verified core and the frontend that must be trusted.

CoCon's Architecture



CoCon’s Verification

Information does not leak from CoCon’s kernel.

CoCon’s Verification

Information does not leak from CoCon’s kernel.

A user learns nothing about a paper’s content beyond the last submitted version unless they become an author of the paper.

CoCon's Verification

Information does not leak from CoCon's kernel.

A user learns nothing about a paper's content beyond the last submitted version unless they become an author of the paper.

Source of Secrets   Observations   Bound Trigger

CoCon's Verification

Information does not leak from CoCon's kernel.

A user learns nothing about a paper's content beyond the last submitted version unless they become an author of the paper.

Source of Secrets   Observations   Bound Trigger

Bounded Deducibility (BD) Security

O can learn nothing about **S** beyond B unless **T** occurs.

1. Introduction
2. **Bounded-Deducibility (BD) Security**
3. CoCon's Verification
4. CoCon in the Wild
5. Other Applications and Developments of BD Security

Bounded-Deducibility (BD) Security

SysTrace $\subseteq$ List(Event)

SysTrace

List(Sec)

List(Obs)

Bounded-Deducibility (BD) Security

SysTrace ⊆ List(Event)

$$\text{isSec} : \text{Event} \to \text{Bool} \qquad \text{getSec} : \text{Event} \to \text{Sec}$$

S = "filter with isSec, then map getSec"

$$\begin{tikzcd}
\text{SysTrace} \arrow[r, "S"] & \text{List(Sec)} \\
\text{List(Obs)} &
\end{tikzcd}$$

Bounded-Deducibility (BD) Security

$\mathsf{SysTrace} \subseteq \mathsf{List}(\mathsf{Event})$

$$\mathsf{isObs} : \mathsf{Event} \to \mathsf{Bool} \qquad \mathsf{getObs} : \mathsf{Event} \to \mathsf{Obs}$$

$$\mathsf{O} = \text{"filter with isObs, then map getObs"}$$

$$\begin{tikzcd}
\mathsf{SysTrace} \arrow[r, "\mathsf{S}"] \arrow[d, "\mathsf{O}"'] & \mathsf{List}(\mathsf{Sec}) \\
\mathsf{List}(\mathsf{Obs}) &
\end{tikzcd}$$

Bounded-Deducibility (BD) Security

$\mathsf{SysTrace} \subseteq \mathsf{List}(\mathsf{Event})$

$\mathsf{T} : \mathsf{Event} \to \mathsf{Bool}$

$$\begin{tikzcd}
\neg \mathsf{T} & \mathsf{SysTrace} \arrow[r, "\mathsf{S}"] \arrow[d, "\mathsf{O}"'] & \mathsf{List}(\mathsf{Sec}) \\
& \mathsf{List}(\mathsf{Obs}) &
\end{tikzcd}$$

Bounded-Deducibility (BD) Security

$\mathsf{SysTrace} \subseteq \mathsf{List(Event)}$

$$\mathsf{T} : \mathsf{Event} \to \mathsf{Bool} \qquad \mathsf{B} : \mathsf{List(Sec)} \to \mathsf{List(Sec)} \to \mathsf{Bool}$$

$$\begin{tikzcd}
\neg \mathsf{T} & \mathsf{SysTrace} \arrow[r, "\mathsf{S}"] \arrow[d, "\mathsf{O}"'] & \mathsf{List(Sec)} \\
 & \mathsf{List(Obs)} &
\end{tikzcd}$$

Bounded-Deducibility (BD) Security

$\mathsf{SysTrace} \subseteq \mathsf{List}(\mathsf{Event})$

Unless T occurs, O can learn nothing about S beyond B

$$\begin{tikzcd}
\neg \mathsf{T} \quad \mathsf{SysTrace} \arrow[rr, "\mathsf{S}"] \arrow[d, "\mathsf{O}"'] & & \mathsf{List}(\mathsf{Sec}) \\
\mathsf{List}(\mathsf{Obs}) & &
\end{tikzcd}$$

Bounded-Deducibility (BD) Security

$\mathrm{SysTrace} \subseteq \mathrm{List(Event)}$

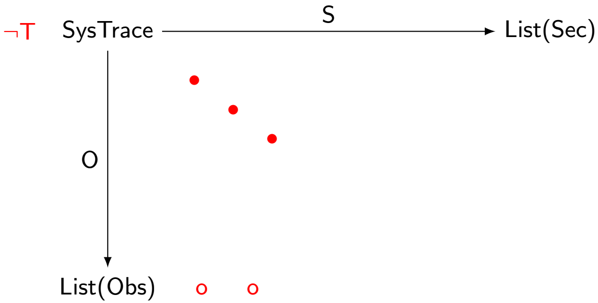
Unless T occurs, O can learn nothing about S beyond B

$$\begin{tikzcd}
\neg \mathrm{T} & \mathrm{SysTrace} \arrow[r, "\mathrm{S}"] \arrow[d, "\mathrm{O}"'] \arrow[dr, phantom, "\ddots"] & \mathrm{List(Sec)} \\
& \mathrm{List(Obs)} &
\end{tikzcd}$$

Bounded-Deducibility (BD) Security

SysTrace $\subseteq$ List(Event)

Unless T occurs, O can learn nothing about S beyond B



Bounded-Deducibility (BD) Security

$\mathsf{SysTrace} \subseteq \mathsf{List}(\mathsf{Event})$

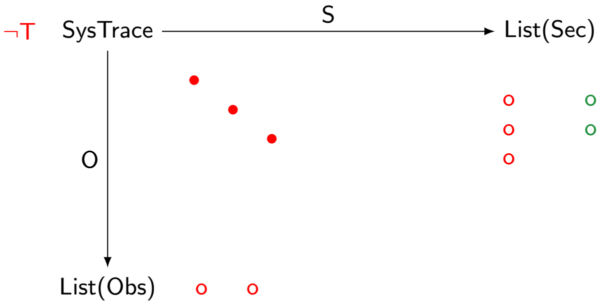
Unless T occurs, O can learn nothing about S beyond B

$$\begin{tikzcd}
\neg \mathsf{T} & \mathsf{SysTrace} \arrow[r, "\mathsf{S}"] \arrow[d, "\mathsf{O}"'] & \mathsf{List}(\mathsf{Sec}) \\
& \mathsf{List}(\mathsf{Obs}) &
\end{tikzcd}$$

Bounded-Deducibility (BD) Security

$\text{SysTrace} \subseteq \text{List}(\text{Event})$

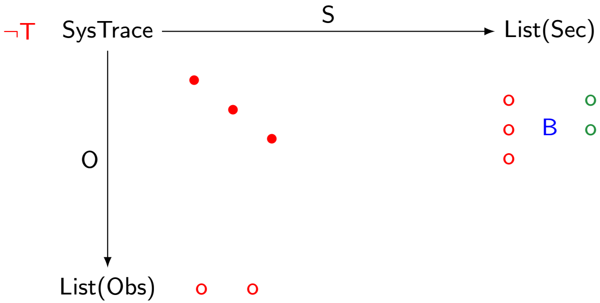
Unless T occurs, O can learn nothing about S beyond B



Bounded-Deducibility (BD) Security

SysTrace $\subseteq$ List(Event)

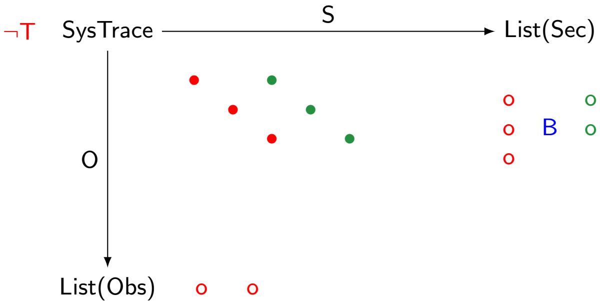
Unless T occurs, O can learn nothing about S beyond B



Bounded-Deducibility (BD) Security

SysTrace ⊆ List(Event)

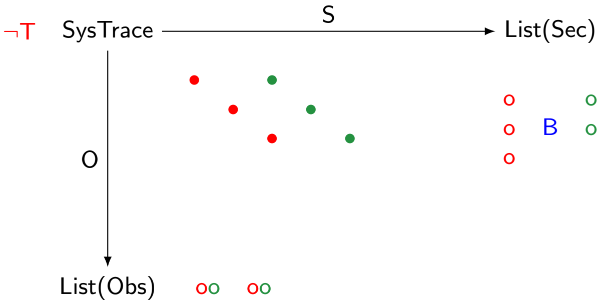
Unless T occurs, O can learn nothing about S beyond B



Bounded-Deducibility (BD) Security

SysTrace ⊆ List(Event)

Unless T occurs, O can learn nothing about S beyond B



¬T SysTrace

S

List(Sec)

O

B

List(Obs)

Formal Definition of BD Security

A system $\mathcal{A}$ is an input/output (I/O) automaton.

An event (or transition) is a quadruple state-input-output-newState.

Formal Definition of BD Security

A system $\mathcal{A}$ is an input/output (I/O) automaton.
An event (or transition) is a quadruple state-input-output-newState.

A flow policy $\mathcal{F}$ for $\mathcal{A}$ consists of:

- an observation infrastructure $(\mathsf{Obs}, \mathsf{isObs}, \mathsf{getObs})$
- a secrecy infrastructure $(\mathsf{Sec}, \mathsf{isSec}, \mathsf{getSec})$
- a declassification bound $\mathsf{B} : \mathsf{List}(\mathsf{Sec}) \to \mathsf{List}(\mathsf{Sec}) \to \mathsf{Bool}$
- a declassification trigger $\mathsf{T} : \mathsf{Event} \to \mathsf{Bool}$

Let $\mathsf{O} = \mathsf{mapfilter\ getObs\ isObs}$ and $\mathsf{S} = \mathsf{mapfilter\ getSec\ isSec}$

Formal Definition of BD Security

A system $\mathcal{A}$ is an input/output (I/O) automaton.
An event (or transition) is a quadruple state-input-output-newState.

A flow policy $\mathcal{F}$ for $\mathcal{A}$ consists of:

- an observation infrastructure $(\mathsf{Obs}, \mathsf{isObs}, \mathsf{getObs})$
- a secrecy infrastructure $(\mathsf{Sec}, \mathsf{isSec}, \mathsf{getSec})$
- a declassification bound $\mathsf{B} : \mathsf{List}(\mathsf{Sec}) \to \mathsf{List}(\mathsf{Sec}) \to \mathsf{Bool}$
- a declassification trigger $\mathsf{T} : \mathsf{Event} \to \mathsf{Bool}$

Let $\mathsf{O} = \mathsf{mapfilter\ getObs\ isObs}$ and $\mathsf{S} = \mathsf{mapfilter\ getSec\ isSec}$

$\mathcal{A}$ being BD secure w.r.t. $\mathcal{F}$, written $\mathcal{A} \models \mathcal{F}$, means:
For all $tr_1 \in \mathsf{SysTrace}$ and $\mathit{sl}_1, \mathit{sl}_2 \in \mathsf{List}(\mathsf{Sec})$,

if never $\mathsf{T}\ tr_1$, $\mathsf{S}\ tr_1 = \mathit{sl}_1$ and $\mathsf{B}\ \mathit{sl}_1\ \mathit{sl}_2$,

then there exists $tr_2 \in \mathsf{SysTrace}$ with $\mathsf{O}\ tr_2 = \mathsf{O}\ tr_1$ and $\mathsf{S}\ tr_1 = \mathit{sl}_1$.

(where never $\mathsf{T}\ tr_1$ means “T holds for no event in tr_1”)

How to Prove BD Security

O can learn nothing about **S** beyond **B** unless **T** occurs

How to Prove BD Security

$\boxed{\textsf{\textbf{O} can learn nothing about \textbf{S} beyond \textbf{B} unless \textbf{T} occurs}}$

Given system trace tr_1 with $\neg\mathsf{T}(tr_1)$ and alternative list of secret secrets sl_2 (within B) ...

How to Prove BD Security

$\boxed{\textbf{O} \text{ can learn nothing about } \textbf{S} \text{ beyond } \textbf{B} \text{ unless } \textbf{T} \text{ occurs}}$

Given system trace tr_1 with $\neg\mathsf{T}(tr_1)$ and alternative list of secret secrets sl_2 (within B) ... exhibit another system trace tr_2 such that

$$\mathsf{O}(tr_2) = \mathsf{O}(tr_1)$$

$$\mathsf{S}(tr_2) = \mathit{sl}_2$$

How to Prove BD Security

O can learn nothing about **S** beyond **B** unless **T** occurs

Given system trace tr_1 with $\neg\mathsf{T}(tr_1)$ and alternative list of secret secrets *sl*$_2$ (within B) ... exhibit another system trace tr_2 such that

$\mathsf{O}(tr_2) = \mathsf{O}(tr_1)$

$\mathsf{S}(tr_2) = \mathit{sl}_2$

Unwinding = Strategy for building tr_2 from tr_1 incrementally

How to Prove BD Security

$\boxed{\textsf{O can learn nothing about S beyond B unless T occurs}}$

Given system trace tr_1 with $\neg\mathsf{T}(tr_1)$ and alternative list of secret secrets sl_2 (within B) ... exhibit another system trace tr_2 such that

$\mathsf{O}(tr_2) = \mathsf{O}(tr_1)$

$\mathsf{S}(tr_2) = \mathit{sl}_2$

Unwinding = Strategy for building tr_2 from tr_1 incrementally

Traditional unwinding: safety-like property

How to Prove BD Security

O can learn nothing about **S** beyond **B** unless **T** occurs

Given system trace tr_1 with $\neg\mathsf{T}(tr_1)$ and alternative list of secret secrets *sl*$_2$ (within B) ... exhibit another system trace tr_2 such that

$\mathsf{O}(tr_2) = \mathsf{O}(tr_1)$

$\mathsf{S}(tr_2) = \mathit{sl}_2$

Unwinding = Strategy for building tr_2 from tr_1 incrementally

Traditional unwinding: safety-like property

Unwinding for BD security: safety + liveness

How to Prove BD Security

Proof by unwinding

$$\begin{tikzcd}
\neg \mathsf{T} & \mathsf{SysTrace} \arrow[rr, "\mathsf{S}"] \arrow[dd, "\mathsf{O}"'] & & \mathsf{List(Sec)} \\
& & & \begin{array}{ccc} \mathsf{o} & & \mathsf{o} \\ \mathsf{o} & \mathsf{B} & \mathsf{o} \\ \mathsf{o} & & \end{array} \\
& \mathsf{List(Obs)} & &
\end{tikzcd}$$

How to Prove BD Security

Proof by unwinding

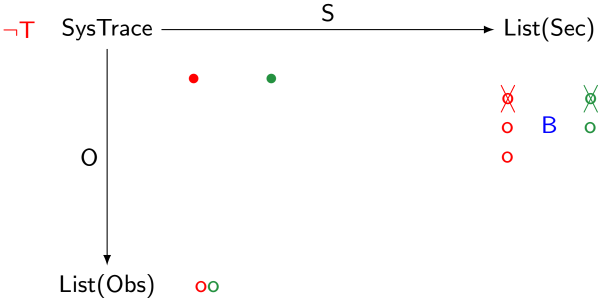
Action

$$\begin{tikzcd}
\neg T & \text{SysTrace} \arrow[rrrr, "S"] \arrow[dddd, "O"'] & & & & \text{List(Sec)} & & \\
& & \bullet & & & & & \\
& & & & & \cancel{o} & & o \\
& & & & & o & B & o \\
& \text{List(Obs)} & o & & & o & &
\end{tikzcd}$$

How to Prove BD Security

Proof by unwinding

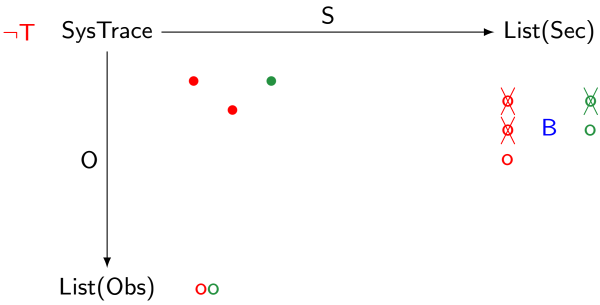
Action / Reaction: Match



How to Prove BD Security

Proof by unwinding

Action



¬T

SysTrace

S

List(Sec)

O

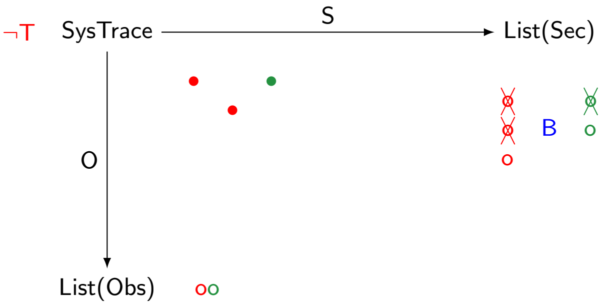
List(Obs)

B

How to Prove BD Security

Proof by unwinding

Action / Reaction: Ignore



¬T

SysTrace

S

List(Sec)

O

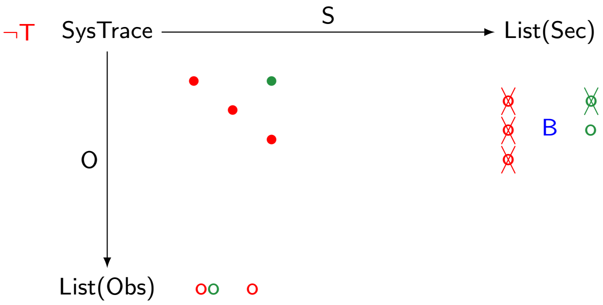
B

List(Obs)

How to Prove BD Security

Proof by unwinding

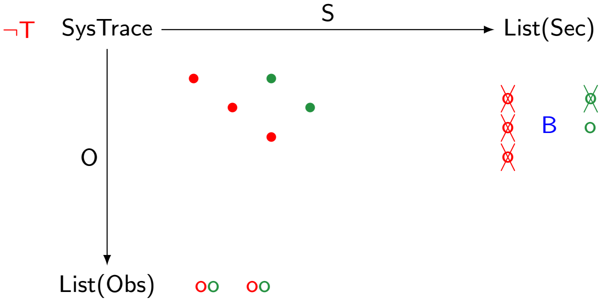
Action



How to Prove BD Security

Proof by unwinding

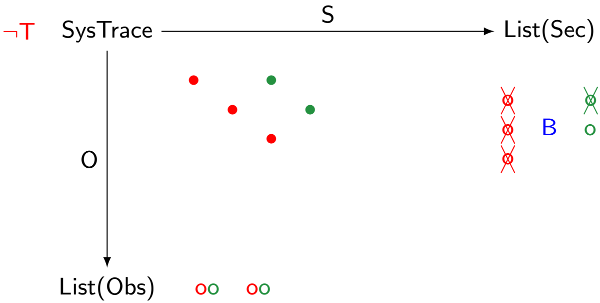
Action / Reaction: Match



How to Prove BD Security

Proof by unwinding

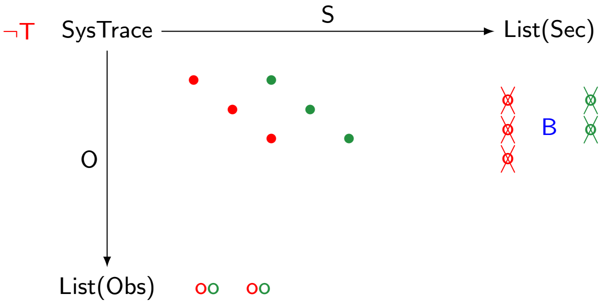
Independent action . . .



How to Prove BD Security

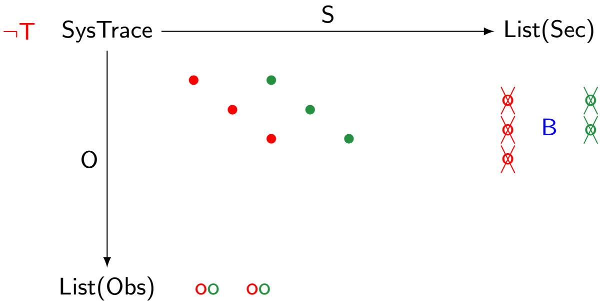
Proof by unwinding

Independent action



How to Prove BD Security

Proof by unwinding ✓



¬T SysTrace

S

List(Sec)

O

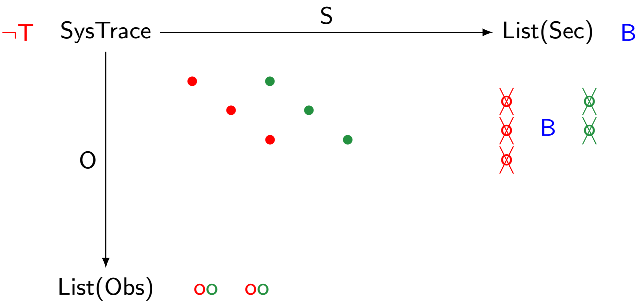
B

List(Obs)

How to Prove BD Security

Proof by unwinding ✓

$$B \mapsto \Delta : \mathsf{State} \times \mathsf{List(Sec)} \times \mathsf{State} \times \mathsf{List(Sec)} \to \mathsf{Bool}$$



Proof by Unwinding

$\Delta : \mathsf{State} \times \mathsf{List}(\mathsf{Sec}) \times \mathsf{State} \times \mathsf{List}(\mathsf{Sec}) \to \mathsf{Bool}$

Proof by Unwinding

Δ : State $\times$ List(Sec) $\times$ State $\times$ List(Sec) $\rightarrow$ Bool

$+$

Strategy for:

- when to act independently
- when to react
- if react: when to match and when to ignore

Proof by Unwinding

$\Delta : \text{State} \times \text{List(Sec)} \times \text{State} \times \text{List(Sec)} \to \text{Bool}$

+

Strategy for:

when to act independently

when to react

if react: when to match and when to ignore

Managing proof complexity:

split unwinding relation into components

have “error” component to exit the unwinding game ASAP

Proof by Unwinding

Δ : State × List(Sec) × State × List(Sec) → Bool

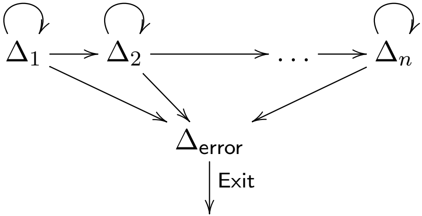
+

Strategy for:

- when to act independently
- when to react
- if react: when to match and when to ignore

Managing proof complexity:

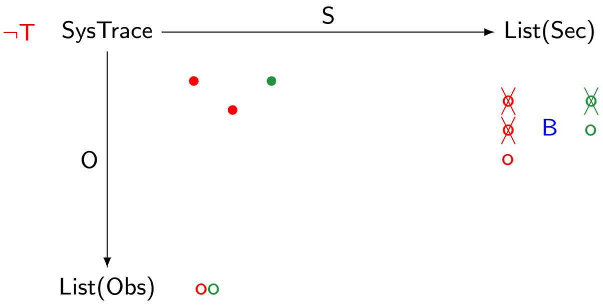
- split unwinding relation into components
- have "error" component to exit the unwinding game ASAP



How to Prove BD Security

Proof by unwinding

Action / Reaction: Ignore



¬T SysTrace

S

List(Sec)

O

List(Obs)

B

Bounded-Deducibility (BD) Security

Takes an Epistemic (Knowledge) Logic perspective to information flow

Generalizes Nondeducibility (Sutherland 1986)

Related notions: Secrecy Maintenance (Halpern and O'Neill 2008) and Gradual Release (Askarov & Sabelfeld 2007)

BD Unwinding generalizes the standard unwinding proof method (Goguen & Meseguer 1984, Mantel 2003)

1. Introduction

2. Bounded-Deducibility (BD) Security

3. **CoCon's Verification**

4. CoCon in the Wild

5. Other Applications and Developments of BD Security

CoCon's Verified Confidentiality Properties

All formulated as instances of BD Security

All verified using BD Unwinding

Secret	Trigger	Bound
Paper Content	Paper Authorship	Last Uploaded Version
	Paper Authorship or PC Membership^B	Nothing
Review	Review Authorship	Last Edited Version Before Discussion and All the Later Versions
	Review Authorship or Non-Conflict PC Membership^D	Last Edited Version Before Notification
	Review Authorship or Non-Conflict PC Membership^D or Paper Authorship^N	Nothing
Discussion	Non-Conflict PC Membership	Nothing
Decision	Non-Conflict PC Membership	Last Edited Version
	Non-Conflict PC Membership or PC Membership^N or Paper Authorship^N	Nothing
Reviewer Assignment to Paper	Non-Conflict PC Membership^R	Non-Conflict PC Membership of Reviewers
	Non-Conflict PC Membership^R or Paper Authorship^N	Non-Conflict PC Membership of Reviewers

Phase Stamps: B = Bidding, D = Discussion, N = Notification, R = Review

Secret	Trigger	Bound
Paper Content	Paper Authorship	Last Uploaded Version
	Paper Authorship or PC Membership^{B}	Nothing
Review	Review Authorship	Last Edited Version Before Discussion and All the Later Versions
	Review Authorship or Non-Conflict PC Membership^{D}	Last Edited Version Before Notification
	Review Authorship or Non-Conflict PC Membership^{D} or Paper Authorship^{N}	Nothing
Discussion	Non-Conflict PC Membership	Nothing
Decision	Non-Conflict PC Membership	Last Edited Version
	Non-Conflict PC Membership or PC Membership^{N} or Paper Authorship^{N}	Nothing
Reviewer Assignment to Paper	Non-Conflict PC Membership^{R}	Non-Conflict PC Membership of Reviewers
	Non-Conflict PC Membership^{R} or Paper Authorship^{N}	Non-Conflict PC Membership of Reviewers

Phase Stamps: B = Bidding, D = Discussion, N = Notification, R = Review

Secret	**Trigger**	**Bound**
Paper Content	Paper Authorship	Last Uploaded Version
	Paper Authorship or PC Membership$^{\mathrm{B}}$	Nothing
Review	Review Authorship	Last Edited Version Before Discussion and All the Later Versions
	Review Authorship or Non-Conflict PC Membership$^{\mathrm{D}}$	Last Edited Version Before Notification
	Review Authorship or Non-Conflict PC Membership$^{\mathrm{D}}$ or Paper Authorship$^{\mathrm{N}}$	Nothing
Discussion	Non-Conflict PC Membership	Nothing
Decision	Non-Conflict PC Membership	Last Edited Version
	Non-Conflict PC Membership or PC Membership$^{\mathrm{N}}$ or Paper Authorship$^{\mathrm{N}}$	Nothing
Reviewer Assignment to Paper	Non-Conflict PC Membership$^{\mathrm{R}}$	Non-Conflict PC Membership of Reviewers
	Non-Conflict PC Membership$^{\mathrm{R}}$ or Paper Authorship$^{\mathrm{N}}$	Non-Conflict PC Membership of Reviewers

Phase Stamps: B = Bidding, D = Discussion, N = Notification, R = Review

Secret	Trigger	Bound
Paper Content	Paper Authorship	Last Uploaded Version
	Paper Authorship or PC Membership^{B}	Nothing
Review	Review Authorship	Last Edited Version Before Discussion and All the Later Versions
	Review Authorship or Non-Conflict PC Membership^{D}	Last Edited Version Before Notification
	Review Authorship or Non-Conflict PC Membership^{D} or Paper Authorship^{N}	Nothing
Discussion	Non-Conflict PC Membership	Nothing
Decision	Non-Conflict PC Membership	Last Edited Version
	Non-Conflict PC Membership or PC Membership^{N} or Paper Authorship^{N}	Nothing
Reviewer Assignment to Paper	Non-Conflict PC Membership^{R}	Non-Conflict PC Membership of Reviewers
	Non-Conflict PC Membership^{R} or Paper Authorship^{N}	Non-Conflict PC Membership of Reviewers

Phase Stamps: B = Bidding, D = Discussion, N = Notification, R = Review

Secret	**Trigger**	**Bound**
Paper Content	Paper Authorship	Last Uploaded Version
	Paper Authorship or PC Membership$^{\mathrm{B}}$	Nothing
Review	Review Authorship	Last Edited Version Before Discussion and All the Later Versions
	Review Authorship or Non-Conflict PC Membership$^{\mathrm{D}}$	Last Edited Version Before Notification
	Review Authorship or Non-Conflict PC Membership$^{\mathrm{D}}$ or Paper Authorship$^{\mathrm{N}}$	Nothing
Discussion	Non-Conflict PC Membership	Nothing
Decision	Non-Conflict PC Membership	Last Edited Version
	Non-Conflict PC Membership or PC Membership$^{\mathrm{N}}$ or Paper Authorship$^{\mathrm{N}}$	Nothing
Reviewer Assignment to Paper	Non-Conflict PC Membership$^{\mathrm{R}}$	Non-Conflict PC Membership of Reviewers
	Non-Conflict PC Membership$^{\mathrm{R}}$ or Paper Authorship$^{\mathrm{N}}$	Non-Conflict PC Membership of Reviewers

Phase Stamps: B = Bidding, D = Discussion, N = Notification, R = Review

Secret	Trigger	Bound
Paper Content	Paper Authorship	Last Uploaded Version
	Paper Authorship or PC Membership^{B}	Nothing
Review	Review Authorship	Last Edited Version Before Discussion and All the Later Versions
	Review Authorship or Non-Conflict PC Membership^{D}	Last Edited Version Before Notification
	Review Authorship or Non-Conflict PC Membership^{D} or Paper Authorship^{N}	Nothing
Discussion	Non-Conflict PC Membership	Nothing
Decision	Non-Conflict PC Membership	Last Edited Version
	Non-Conflict PC Membership or PC Membership^{N} or Paper Authorship^{N}	Nothing
Reviewer Assignment to Paper	Non-Conflict PC Membership^{R}	Non-Conflict PC Membership of Reviewers and Number of Reviewers
	Non-Conflict PC Membership^{R} or Paper Authorship^{N}	Non-Conflict PC Membership of Reviewers

Phase Stamps: B = Bidding, D = Discussion, N = Notification, R = Review

Overall Verification Effort

BD Security framework: 1000 LOC

Confidentiality properties: 5000 LOC

Safety properties: 1000 LOC

Traceback properties: 700 LOC

Overall Verification Effort

BD Security framework: 1000 LOC

Confidentiality properties: 5000 LOC

1. Only non-conflict PC members may learn such and such

Safety properties: 1000 LOC

Traceback properties: 700 LOC

Overall Verification Effort

BD Security framework: 1000 LOC

Confidentiality properties: 5000 LOC

1. Only non-conflict PC members may learn such and such

Safety properties: 1000 LOC

2. And authors are always in conflict with their papers

Traceback properties: 700 LOC

Overall Verification Effort

BD Security framework: 1000 LOC

Confidentiality properties: 5000 LOC

1. Only non-conflict PC members may learn such and such

Safety properties: 1000 LOC

2. And authors are always in conflict with their papers

(1) + (2) $\longrightarrow$ Authors never learn such and such

Traceback properties: 700 LOC

Overall Verification Effort

BD Security framework: 1000 LOC

Confidentiality properties: 5000 LOC

Safety properties: 1000 LOC

Traceback properties: 700 LOC

Traceback Properties

We proved: One cannot learn beyond such an such unless one is or becomes such and such

Traceback Properties

We proved: One cannot learn beyond such an such unless one is or becomes such and such

But how can one become such and such?

E.g., how could 👤 become an author of 📄 ?

Traceback Properties

We proved: One cannot learn beyond such an such unless one is or becomes such and such

But how can one become such and such?

E.g., how could 👤 become an author of 📄 ?

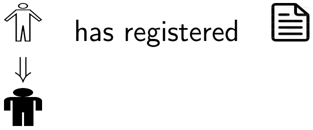
👤 has registered 📄

Traceback Properties

We proved: One cannot learn beyond such an such unless one is or becomes such and such

But how can one become such and such?

E.g., how could 🧍 become an author of 📄 ?

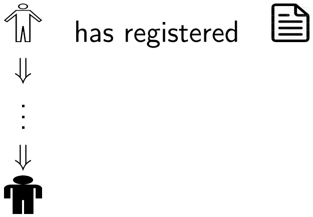


Traceback Properties

We proved: One cannot learn beyond such an such unless one is or becomes such and such

But how can one become such and such?

E.g., how could 🧍 become an author of 📄 ?



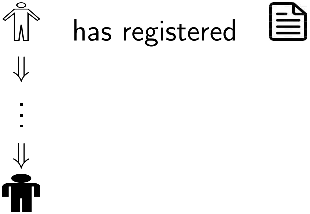
has registered

Traceback Properties

We proved: One cannot learn beyond such an such unless one is or becomes such and such

But how can one become such and such?

E.g., how could 👤 become an author of 📄 ?



Confidentiality + Safety + Traceback $\Longrightarrow$ Relax

Some Trivia

One reviewer's reaction to our submission describing CoCon's verification:

> "The authors cunningly chose a topic that
> directly speaks to the reviewers of their paper."

Some Trivia

One reviewer's reaction to our submission describing CoCon's verification:

- "The authors cunningly chose a topic that directly speaks to the reviewers of their paper."

Andrei Voronkov's assessment of (a preliminary version of) CoCon:

- "A true Mickey Mouse system!"

Some Trivia

One reviewer's reaction to our submission describing CoCon's verification:

> "The authors cunningly chose a topic that directly speaks to the reviewers of their paper."

Andrei Voronkov's assessment of (a preliminary version of) CoCon:

> "A true Mickey Mouse system!"

CoCon has a superuser called "The Voronkov" to acknowledge CoCon's inspiration from EasyChair.

1. Introduction
2. Bounded-Deducibility (BD) Security
3. CoCon's Verification
4. **CoCon in the Wild**
5. Other Applications and Developments of BD Security

CoCon in the Wild

CoCon has been used to manage two international conferences:

TABLEAUX 2015: The 24th Conference on Automated Reasoning with Analytic Tableaux and Related Methods

ITP 2016: The 7th Conference on Interactive Theorem Proving (29th if we count its predecessor conference)

CoCon at TABLEAUX 2015

Hans de Nivelle
conference chair

CoCon at TABLEAUX 2015

Hans de Nivelle
conference chair

CoCon?? At TABLEAUX 2015?? Not a chance!

CoCon at TABLEAUX 2015

Hans de Nivelle
conference chair

CoCon?? At TABLEAUX 2015?? Not a chance!

But maybe CoCon++. . .

CoCon++

CoCon plus the following features:

Various convenience listings, e.g.:

- For PC members: papers listed by average score
- For the chair: paper load of each PC, reviewer number for each paper

Email notifications:

- to authors about the decision
- to PC members about the addition of comments or new reviews to their reviewed papers

CoCon++

CoCon plus the following features:

Various convenience listings, e.g.:

- For PC members: papers listed by average score
- For the chair: paper load of each PC, reviewer number for each paper

Email notifications:

- to authors about the decision
- to PC members about the addition of comments or new reviews to their reviewed papers

Is CoCon++ as secure as CoCon?

CoCon++

CoCon plus the following features:

Various convenience listings, e.g.: OK

- For PC members: papers listed by average score
- For the chair: paper load of each PC, reviewer number for each paper

Email notifications:

- to authors about the decision
- to PC members about the addition of comments or new reviews to their reviewed papers

Is CoCon++ as secure as CoCon?

CoCon++

CoCon plus the following features:

Various convenience listings, e.g.: OK

- For PC members: papers listed by average score
- For the chair: paper load of each PC, reviewer number for each paper

Email notifications: OK, if done right

- to authors about the decision
- to PC members about the addition of comments or new reviews to their reviewed papers

Is CoCon++ as secure as CoCon?

CoCon++

CoCon plus the following features:

Various convenience listings, e.g.: OK

- For PC members: papers listed by average score
- For the chair: paper load of each PC, reviewer number for each paper

Email notifications: OK, if done right

- to authors about the decision
- to PC members about the addition of comments or new reviews to their reviewed papers

Is CoCon++ as secure as CoCon?

We proved that it is!

On the way: designed framework for secure system extensions.

CoCon at TABLEAUX 2015

Had proved about CoCon's kernel:
An author learns nothing about the score of their paper before notification.

TABLEAUX
PC member in
Discussion phase

I'm also an author and I see my paper listed somewhere in the middle.
Does it mean I can already infer something about its score?

CoCon at TABLEAUX 2015

Had proved about CoCon's kernel:
An author learns nothing about the score of their paper before notification.

TABLEAUX
PC member in
Discussion phase

I'm also an author and I see my paper listed somewhere in the middle. Does it mean I can already infer something about its score?

No, that was only a pseudo-leak.

Web interface written around the verified kernel – web developer decided to treat score "unknown" as 0 when sorting papers by score to display to a user.

CoCon at ITP 2016

Jasmin Blanchette
conference co-chair

Announcement: The 7th International Conference on Interactive Theorem Proving
22 to 26 August 2016, Nancy, France
...

Papers should be submitted in PDF format via EasyChair.

CoCon at ITP 2016

Jasmin Blanchette
conference co-chair

Announcement: The 7th International Conference on Interactive Theorem Proving 22 to 26 August 2016, Nancy, France

...

Papers should be submitted in PDF format via EasyChair.

My email message to Jasmin: Traitor!

CoCon at ITP 2016

Jasmin Blanchette
conference co-chair

Announcement: The 7th International Conference on Interactive Theorem Proving
22 to 26 August 2016, Nancy, France
...

Papers should be submitted in PDF format via EasyChair.

My email message to Jasmin: Traitor!
Jasmin: I know this will sound crazy, but the reason we're using EasyChair is simply that I forgot!!

CoCon at ITP 2016

Jasmin Blanchette
conference co-chair

Announcement: The 7th International Conference on Interactive Theorem Proving
22 to 26 August 2016, Nancy, France
...

Papers should be submitted in PDF format via EasyChair.

My email message to Jasmin: Traitor!

Jasmin: I know this will sound crazy, but the reason we're using EasyChair is simply that I forgot!!

Me: No hard feelings; in fact, it would've been a huge amount of stress for me.

CoCon at ITP 2016

Jasmin Blanchette
conference co-chair

Announcement: The 7th International Conference on Interactive Theorem Proving
22 to 26 August 2016, Nancy, France
...

Papers should be submitted in PDF format via EasyChair.

My email message to Jasmin: Traitor!

Jasmin: I know this will sound crazy, but the reason we're using EasyChair is simply that I forgot!!

Me: No hard feelings; in fact, it would've been a huge amount of stress for me.

Jasmin: In that case, it could be fun. Maybe we *should* go with CoCon.

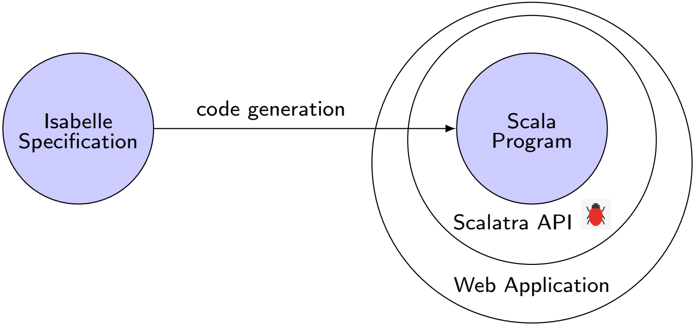
CoCon at ITP 2016

Andrew Tolmach
PC member

I'm a little nervous that
some unintended leakage
may be occurring!
... why did I see the scores
of a paper I have conflict with?

What Had Happened?

CoCon's Architecture



Critical Data Race Bug Outside the Verified Kernel

Why have we not discovered it

- during heavy testing
- during countless conference simulations
- during TABLEAUX 2015

Critical Data Race Bug Outside the Verified Kernel

Why have we not discovered it

- during heavy testing
- during countless conference simulations
- during TABLEAUX 2015

Because it occurs very seldom

- Needs sufficient delays (caused by high traffic)
- Is volatile: happens per single request, then vanishes

Critical Data Race Bug Outside the Verified Kernel

Why have we not discovered it

- during heavy testing
- during countless conference simulations
- during TABLEAUX 2015 – 70 users

  ITP 2016 – 110 users

Because it occurs very seldom

- Needs sufficient delays (caused by high traffic)
- Is volatile: happens per single request, then vanishes

What were the consequences?

What we verified for the kernel
was compromised by the API layer.

What were the consequences?

What we verified for the kernel
was compromised by the API layer.

However

- Due to rarity and volatilily, users would not notice
- Really sensitive data was at least two clicks away
- Authors and PC members were accessing the system at different times

What were the consequences?

What we verified for the kernel
was compromised by the API layer.

However

- Due to rarity and volatilily, users would not notice
- Really sensitive data was at least two clicks away
- Authors and PC members were accessing the system at different times

The bug was fixed during the discussion phase.

Andrew Tolmach:

IMO, the whole episode is an unusually clear illustration of the perils of overselling (even to oneself) the benefits of verification. The most interesting thing is not that CoCon had a bug, but rather that the developers were (temporarily) in denial about it. The most revealing pieces of the email exchange over the bug were the following from Andrei in response to my bug report:

> A leak is impossible this way: such information does not leak through the kernel.

and a little later:

> Again, I am not sure what is going on, but I am sure that it is not a leak. The server is accessed with your credentials, and the scores of reviews of conflicted papers are not accessible with your credentials.

Lessons Learned

Unverified part deserves special attention

- Large effort verifying the kernel's rich information flow
- Not enough effort in reviewing carefully the thin outer layer

Lessons Learned

Unverified part deserves special attention

- Large effort verifying the kernel's rich information flow
- Not enough effort in reviewing carefully the thin outer layer

Some discussion on extending CoCon's verification to the outer layers.

Lessons Learned

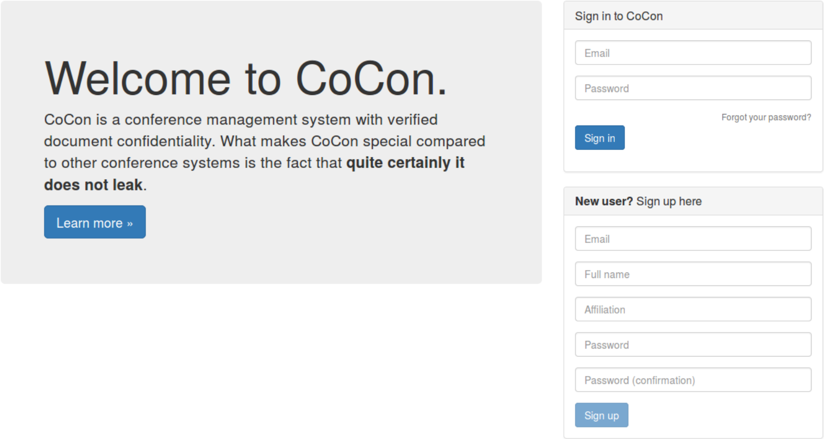
Unverified part deserves special attention

- Large effort verifying the kernel's rich information flow
- Not enough effort in reviewing carefully the thin outer layer

Some discussion on extending CoCon's verification to the outer layers.

Important to always state precisely which part of the system was verified – even in a summary!

Before ITP 2016



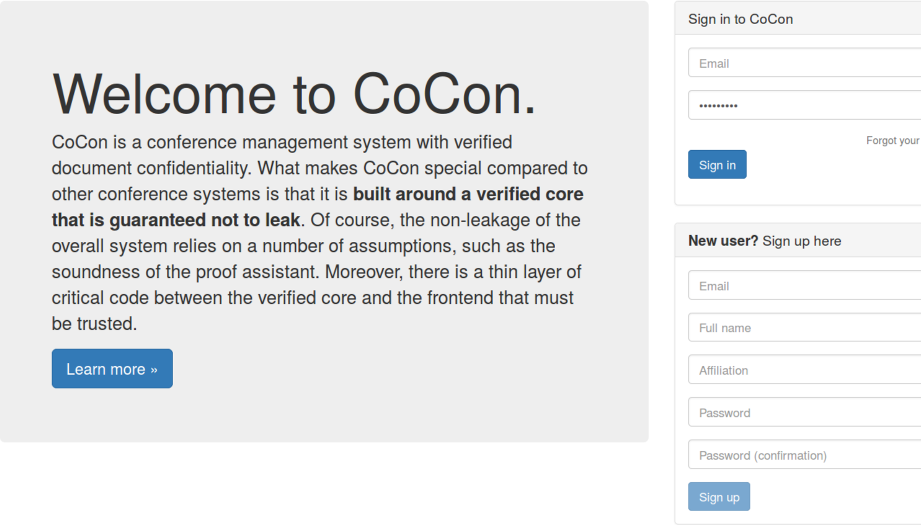
Welcome to CoCon.

CoCon is a conference management system with verified document confidentiality. What makes CoCon special compared to other conference systems is the fact that **quite certainly it does not leak**.

Sign in to CoCon

New user? Sign up here

After ITP 2016



Welcome to CoCon.

CoCon is a conference management system with verified document confidentiality. What makes CoCon special compared to other conference systems is that it is **built around a verified core that is guaranteed not to leak**. Of course, the non-leakage of the overall system relies on a number of assumptions, such as the soundness of the proof assistant. Moreover, there is a thin layer of critical code between the verified core and the frontend that must be trusted.

Learn more »

Sign in to CoCon

Email

•••••••••

Forgot your

Sign in

New user? Sign up here

Email

Full name

Affiliation

Password

Password (confirmation)

Sign up

More Details on CoCon's Verification and Deployment Saga

Andrei Popescu, Peter Lammich, Ping Hou.
CoCon: A Conference Management System with Formally Verified Document Confidentiality.
Journal of Automated Reasoning 2021

1. Introduction
2. Bounded-Deducibility (BD) Security
3. CoCon's Verification
4. CoCon in the Wild
5. **Other Applications and Developments of BD Security**

CoSMed

Facebook-style social media platform

- users register, create posts (which can later be updated) and establish friendship relationships
- post access can be public or friends-only (and this can change)

Implemented using the same scheme as CoCon

CoSMed – Example of Desirable Confidentiality Property

A group of users learn nothing about a post unless one of them is the admin, or is the post's owner, or becomes friends with the owner, or the post gets marked as public.

CoSMed – Example of Desirable Confidentiality Property

A group of users learn nothing about a post unless one of them is the admin, or is the post's owner, or becomes friends with the owner, or the post gets marked as public.

Bound: complete nondeducibility

Trigger: Acquisition of various roles or the opening of access

CoSMed – Example of Desirable Confidentiality Property

A group of users learn nothing about a post unless one of them is the admin, or is the post's owner, or becomes friends with the owner, or the post gets marked as public.

Bound: complete nondeducibility

Trigger: Acquisition of various roles or the opening of access

Too weak

CoSMed – Example of Desirable Confidentiality Property

A group of users learn nothing about a post unless one of them is the admin, or is the post's owner, or becomes friends with the owner, or the post gets marked as public.

Bound: complete nondeducibility

Trigger: Acquisition of various roles or the opening of access

Too weak

Better justice to what's going on (stronger confidentiality property):

A group of users learn about a post nothing beyond the updates performed while (or last before) one of them is the admin, or is the post's owner, or becomes friends with the owner, or the post is marked as public.

CoSMed – Example of Desirable Confidentiality Property

A group of users learn nothing about a post unless one of them is the admin, or is the post's owner, or becomes friends with the owner, or the post gets marked as public.

Bound: complete nondeducibility

Trigger: Acquisition of various roles or the opening of access

Too weak

Better justice to what's going on (stronger confidentiality property):

A group of users learn about a post nothing beyond the updates performed while (or last before) one of them is the admin, or is the post's owner, or becomes friends with the owner, or the post is marked as public.

inductive bound B

CoSMed – Example of Desirable Confidentiality Property

A group of users learn nothing about a post unless one of them is the admin, or is the post's owner, or becomes friends with the owner, or the post gets marked as public.

Bound: complete nondeducibility

Trigger: Acquisition of various roles or the opening of access

Too weak

Better justice to what's going on (stronger confidentiality property):

A group of users learn about a post nothing beyond the updates performed while (or last before) one of them is the admin, or is the post's owner, or becomes friends with the owner, or the post is marked as public.

inductive bound B_T

CoSMed – Proved Confidentiality Properties

Secret	(Trigger-Swallowing) Bound
Content of a given post	Updates performed while or last before one of the following holds: – Some user in G is the admin, is the post owner or is friends with its owner – The post is marked as public
Friendship status between two given users U and V	Status changes performed while or last before the following holds: – Some user in G is the admin or is friends with U or V
Friendship requests between two given users U and V	Existence of accepted requests while or last before the following holds: – Some user in G is the admin or is friends with U or V

The observations are made by a group of users G. The trigger is vacuously false.

Bauereiss, Pesenti Gritti, Popescu, Raimondi. CoSMed: A Confidentiality-Verified Social Media Platform. ITP 2016, JAR 2018.

CoSMeDis

Diaspora-style extension of CosMed:

- multiple CosMed nodes deployed at different sites
- any two nodes can connect: posts can be shared and friendships can be established cross-nodes

CoSMeDis

Diaspora-style extension of CosMed:

- multiple CosMed nodes deployed at different sites
- any two nodes can connect: posts can be shared and friendships can be established cross-nodes

Question: Do CosMed's confidentiality guarantees extend to CoSMeDis?

CoSMeDis

Diaspora-style extension of CosMed:

- multiple CosMed nodes deployed at different sites
- any two nodes can connect: posts can be shared and friendships can be established cross-nodes

Question: Do CosMed's confidentiality guarantees extend to CoSMeDis?

Broader research question: How to compose BD Security flow policies of individual nodes to form guarantees for the entire network?

Compositionality Theorem for BD Security

Rough Statement of the Theorem. If n communicating systems (e.g., n CoSMeDis nodes) have their communication:

- observable to a sufficient degree, and
- secret-polarized (i.e., only of the nodes issues the secrets of interest),

and each of them satisfies a BD security policy $\mathcal{F}_i$, then their communicating product satisfies a naturally defined product policy $\prod_{i=1}^{n} \mathcal{F}_i$.

Compositionality Theorem for BD Security

Rough Statement of the Theorem. If n communicating systems (e.g., n CoSMeDis nodes) have their communication:

- observable to a sufficient degree, and
- secret-polarized (i.e., only of the nodes issues the secrets of interest),

and each of them satisfies a BD security policy $\mathcal{F}_i$, then their communicating product satisfies a naturally defined product policy $\prod_{i=1}^{n} \mathcal{F}_i$.

Strength: Policy agnosticism – compose any policies $\mathcal{F}_i$, no questions asked.

Compositionality Theorem for BD Security

Rough Statement of the Theorem. If n communicating systems (e.g., n CoSMeDis nodes) have their communication:

- observable to a sufficient degree, and
- secret-polarized (i.e., only of the nodes issues the secrets of interest),

and each of them satisfies a BD security policy $\mathcal{F}_i$, then their communicating product satisfies a naturally defined product policy $\prod_{i=1}^{n} \mathcal{F}_i$.

Strength: Policy agnosticism – compose any policies $\mathcal{F}_i$, no questions asked.

Weakness: Restriction on communicating with only one secret source.

Compositionality Theorem for BD Security

Rough Statement of the Theorem. If n communicating systems (e.g., n CoSMeDis nodes) have their communication:

- observable to a sufficient degree, and
- secret-polarized (i.e., only of the nodes issues the secrets of interest),

and each of them satisfies a BD security policy $\mathcal{F}_i$, then their communicating product satisfies a naturally defined product policy $\prod_{i=1}^{n} \mathcal{F}_i$.

Strength: Policy agnosticism – compose any policies $\mathcal{F}_i$, no questions asked.

Weakness: Restriction on communicating with only one secret source.

Applied to lift CoSMed's confidentiality guarantees to CoSMeDis.

Bauereiss, Pesenti Gritti, Popescu, Raimondi. CoSMeDis: A Distributed Social Media Platform with Formally Verified Confidentiality Guarantees. IEEE Symposium on Security and Privacy, 2017

Confidentiality Properties Lifted from CoSMed to CoSMeDis

Secret	Bound
Content of a given post at node i	Updates performed while or last before one of the following holds: – Some user in G_i is the node's admin, is the post owner or is friends with its owner – The post is marked as public – Some user in G_j for $j \neq i$ is the admin at node j or is remote friends with the post's owner
Friendship status between two given users U and V at node i	Status changes performed while or last before the following holds: – Some user at node i is the node's admin or is friends with U or V
Friendship requests between two given users U and V at node i	Existence of accepted requests while or last before the following holds: – Some user at node i is the node's admin or is friends with U or V

The observations are made by n groups of users—one group G_i at each node i. The declassification trigger is again vacuously false.

Summary of BD Security

Framework for expressing and verifying fine-grained information flow security properties

Formalized in Isabelle/HOL

Comes with mechanisms for managing complexity: compositional incremental proof machinery, compositionality results

Fine-tuned on some large verification projects: CoCon, CoSMed, CoSMeDis

Summary of BD Security

Framework for expressing and verifying fine-grained information flow security properties

Formalized in Isabelle/HOL

Comes with mechanisms for managing complexity: compositional incremental proof machinery, compositionality results

Fine-tuned on some large verification projects: CoCon, CoSMed, CoSMeDis

Try it today for free (available from the Isabelle AFP)

More Related/Relevant/Inspiring Work

Systems verified for information-flow security: hardware architecture with information flow primitives (Amorim et al.), an ARM-based separation kernel (Dam et al.), noninterference for seL4 (Murray et al.), the Quark verified browser (Jang et al.)

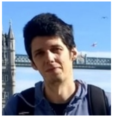
Automatic analysis of information flow security: Jif/Fabric (Myers, Liu et al.), LIO/Hails (Giffin et al.), Paragon (Broberg et al.), Jeeves (Yang at al.) and Ur/Web (Chlipala).

Information Flow Security for Conference Management Systems: ConfiChair (Arapinis et al.), Qapla (Mehta et al.)

Temporal logic approaches: SecLTL (Dimitrova et al.), HyperLTL (Clarkson et al.)

Compositionality results: McCullough's early work, Mantel's MAKS

Bounded-Deducibility Security. ITP 2021



Popescu, Lammich, Hou. CoCon: A Conference Management System with Formally Verified Document Confidentiality. Journal of Automated Reasoning, 2021.

Bauereiss, Pesenti Gritti, Popescu, Raimondi. CoSMed: A Confidentiality-Verified Social Media Platform. Journal of Automated Reasoning, 2018. (Journal version of ITP 2016 paper)

Bauereiss, Pesenti Gritti, Popescu, Raimondi. CoSMeDis: A Distributed Social Media Platform with Formally Verified Confidentiality Guarantees. IEEE Symposium on Security and Privacy, 2017.

Kanav, Lammich, Popescu. A Conference Management System with Verified Document Confidentiality. CAV 2014.

Support from DFG (through RS^3), EPSRC and NCSC/VeTSS gratefully acknowledged.

Reserve Slides

Secret	**Trigger**	**Bound**
Paper Content	Paper Authorship	Last Uploaded Version
	Paper Authorship or PC Membership^{B}	Nothing
Review	Review Authorship	Last Edited Version Before Discussion and All the Later Versions
	Review Authorship or Non-Conflict PC Membership^{D}	Last Edited Version Before Notification
	Review Authorship or Non-Conflict PC Membership^{D} or Paper Authorship^{N}	Nothing
Discussion	Non-Conflict PC Membership	Nothing
Decision	Non-Conflict PC Membership	Last Edited Version
	Non-Conflict PC Membership or PC Membership^{N} or Paper Authorship^{N}	Nothing
Reviewer Assignment to Paper	Non-Conflict PC Membership^{R}	Non-Conflict PC Membership of Reviewers
	Non-Conflict PC Membership^{R} or Paper Authorship^{N}	Non-Conflict PC Membership of Reviewers

Phase Stamps: B = Bidding, D = Discussion, N = Notification, R = Review

Example of Formalizing a Flow Policy

A group of users UIDs learns nothing about the content of a paper's review (say, review N of paper PID) beyond the last submitted version before the discussion phase and the later versions unless one of them is that review's author.

Example of Formalizing a Flow Policy

A group of users UIDs learns nothing about the content of a paper's review (say, review N of paper PID) beyond the last submitted version before the discussion phase and the later versions unless one of them is that review's author.

Observation infrastructure: retain from a trace all pairs input–output for users in UIDs.

Example of Formalizing a Flow Policy

A group of users UIDs learns nothing about the content of a paper's review (say, review N of paper PID) beyond the last submitted version before the discussion phase and the later versions unless one of them is that review's author.

Observation infrastructure: retain from a trace all pairs input–output for users in UIDs.

Secrecy infrastructure: retain from a trace all pairs phase–update, where:
"update" means "update to review N of paper PID", and
"phase" means "the current phase of the conference".

Example of Formalizing a Flow Policy

A group of users UIDs learns nothing about the content of a paper's review (say, review N of paper PID) beyond the last submitted version before the discussion phase and the later versions unless one of them is that review's author.

Observation infrastructure: retain from a trace all pairs input–output for users in UIDs.

Secrecy infrastructure: retain from a trace all pairs phase–update, where:
"update" means "update to review N of paper PID", and
"phase" means "the current phase of the conference".

Trigger: T returns true for a transition iff the transition's target state has a user in UIDs as reviewer for review N of paper PID.

Example of Formalizing a Flow Policy

A group of users UIDs learns nothing about the content of a paper's review (say, review N of paper PID) beyond the last submitted version before the discussion phase and the later versions unless one of them is that review's author.

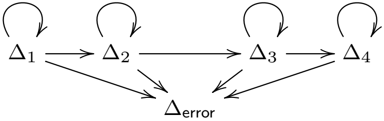
Observation infrastructure: retain from a trace all pairs input–output for users in UIDs.

Secrecy infrastructure: retain from a trace all pairs phase–update, where:
"update" means "update to review N of paper PID", and
"phase" means "the current phase of the conference".

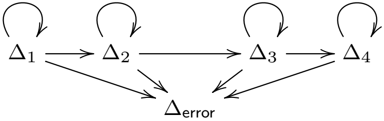
Trigger: T returns true for a transition iff the transition's target state has a user in UIDs as reviewer for review N of paper PID.

Bound: Two lists of secrets, which are lists of pairs phase–update, are related by B iff:
their suffixes consisting of pairs having phase Discussion are equal, and
their last updates before those suffixes are also equal.

Network of Unwinding Relations for Proving It

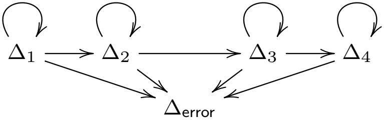


Network of Unwinding Relations for Proving It



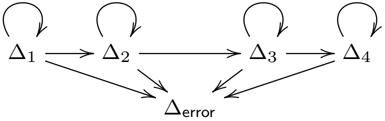
$\Delta_1\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\forall cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \to \text{phase}\ \sigma_1\ cid < \text{Reviewing})\land$ $\sigma_1 = \sigma_2 \land \text{B}\ sl_1\ sl_2$
$\Delta_2\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid = \text{Reviewing} \land$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N})) \land$ $\sigma_1 = \sigma_2 \land \text{B}\ sl_1\ sl_2$
$\Delta_3\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid = \text{Reviewing} \land$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}) \land$ $\sigma_1 =_{\text{PID,N}} \sigma_2 \land \text{B}\ sl_1\ sl_2$
$\Delta_4\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid \geq \text{Reviewing} \land$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}) \land$ $\sigma_1 = \sigma_2 \land (\exists wl.\ sl_1 = sl_2 = \text{map (Pair Discussion)}\ wl)$
$\Delta_{\text{error}}\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$sl_1 \neq [\,] \land$ $((\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid > \text{Reviewing}\land$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N}))$ $\lor$ $(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid > \text{Reviewing} \land$ $\text{fst (hd}\ sl_1) = \text{Reviewing}))$

Network of Unwinding Relations for Proving It



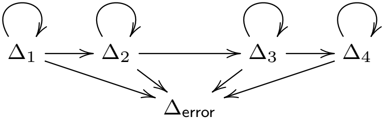
$\Delta_1\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\forall cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \to \text{phase}\ \sigma_1\ cid < \text{Reviewing}) \wedge$ $\sigma_1 = \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_2\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \ \wedge\ \text{phase}\ \sigma_1\ cid = \text{Reviewing} \ \wedge$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N})) \wedge$ $\sigma_1 = \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_3\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid = \text{Reviewing} \wedge$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}\ ) \wedge$ $\sigma_1 =_{\text{PID,N}} \sigma_2\ \wedge \text{B}\ sl_1\ sl_2$
$\Delta_4\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge\ \text{phase}\ \sigma_1\ cid \geq \text{Reviewing}\ \wedge$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}) \wedge$ $\sigma_1 = \sigma_2\ \wedge (\ \exists wl.\ sl_1 = sl_2 = \text{map (Pair Discussion)}\ wl\ )$
$\Delta_{\text{error}}\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$sl_1 \neq [] \wedge$ $((\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid > \text{Reviewing} \wedge$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N}))$ $\vee$ $(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid > \text{Reviewing} \wedge$ $\text{fst (hd}\ sl_1) = \text{Reviewing}))$

Network of Unwinding Relations for Proving It



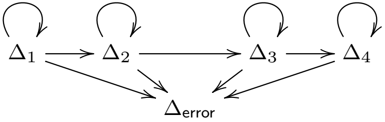
$\Delta_1\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\forall cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \to \text{phase}\ \sigma_1\ cid < \text{Reviewing}) \wedge$ $\sigma_1 = \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_2\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid = \text{Reviewing} \wedge$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N})) \wedge$ $\sigma_1 = \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_3\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid = \text{Reviewing} \wedge$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}\ ) \wedge$ $\sigma_1 =_{\text{PID,N}} \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_4\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid \geq \text{Reviewing} \wedge$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}) \wedge$ $\sigma_1 = \sigma_2 \wedge (\ \exists wl.\ sl_1 = sl_2 = \text{map}\ (\text{Pair Discussion})\ wl\ )$
$\Delta_{\text{error}}\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$sl_1 \neq [] \wedge$ $((\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid > \text{Reviewing} \wedge$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N}))$ $\vee$ $(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid > \text{Reviewing} \wedge$ $\text{fst}\ (\text{hd}\ sl_1) = \text{Reviewing}))$

Network of Unwinding Relations for Proving It



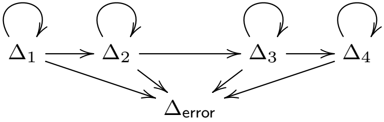
$\Delta_1\ (\sigma_1, \mathit{sl}_1, \sigma_2, \mathit{sl}_2)$	$(\forall cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \to \text{phase}\ \sigma_1\ cid < \text{Reviewing}) \land$ $\sigma_1 = \sigma_2 \land \text{B}\ \mathit{sl}_1\ \mathit{sl}_2$
$\Delta_2\ (\sigma_1, \mathit{sl}_1, \sigma_2, \mathit{sl}_2)$	$(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid = \text{Reviewing} \land$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N})) \land$ $\sigma_1 = \sigma_2 \land \text{B}\ \mathit{sl}_1\ \mathit{sl}_2$
$\Delta_3\ (\sigma_1, \mathit{sl}_1, \sigma_2, \mathit{sl}_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid = \text{Reviewing} \land$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}) \land$ $\sigma_1 =_{\text{PID,N}} \sigma_2 \land \text{B}\ \mathit{sl}_1\ \mathit{sl}_2$
$\Delta_4\ (\sigma_1, \mathit{sl}_1, \sigma_2, \mathit{sl}_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid \geq \text{Reviewing} \land$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}) \land$ $\sigma_1 = \sigma_2 \land (\exists wl.\ \mathit{sl}_1 = \mathit{sl}_2 = \text{map (Pair Discussion)}\ wl)$
$\Delta_{\text{error}}\ (\sigma_1, \mathit{sl}_1, \sigma_2, \mathit{sl}_2)$	$\mathit{sl}_1 \neq [] \land$ $((\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid > \text{Reviewing} \land$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N}))$ $\lor$ $(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \land \text{phase}\ \sigma_1\ cid > \text{Reviewing} \land$ $\text{fst (hd}\ \mathit{sl}_1) = \text{Reviewing}))$

Network of Unwinding Relations for Proving It



$\Delta_1\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\forall cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \rightarrow \text{phase}\ \sigma_1\ cid < \text{Reviewing}) \wedge$ $\sigma_1 = \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_2\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid = \text{Reviewing} \wedge$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N})) \wedge$ $\sigma_1 = \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_3\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid = \text{Reviewing} \wedge$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}) \wedge$ $\sigma_1 =_{\text{PID,N}} \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_4\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid \geq \text{Reviewing} \wedge$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}) \wedge$ $\sigma_1 = \sigma_2 \wedge (\exists wl.\ sl_1 = sl_2 = \text{map}\ (\text{Pair Discussion})\ wl)$
$\Delta_{\text{error}}\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$sl_1 \neq [] \wedge$ $((\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid > \text{Reviewing} \wedge$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N}))$ $\vee$ $(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid > \text{Reviewing} \wedge$ $\text{fst}\ (\text{hd}\ sl_1) = \text{Reviewing}))$

Network of Unwinding Relations for Proving It



$\Delta_1\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\forall cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \rightarrow \text{phase}\ \sigma_1\ cid < \text{Reviewing})\wedge$ $\sigma_1 = \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_2\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid = \text{Reviewing} \wedge$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N})) \wedge$ $\sigma_1 = \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_3\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid = \text{Reviewing} \wedge$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}\ ) \wedge$ $\sigma_1 =_{\text{PID,N}} \sigma_2 \wedge \text{B}\ sl_1\ sl_2$
$\Delta_4\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$(\exists cid\ uid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid \geq \text{Reviewing} \wedge$ $\text{isRevNth}\ \sigma_1\ uid\ \text{PID N}) \wedge$ $\sigma_1 = \sigma_2 \wedge (\ \exists wl.\ sl_1 = sl_2 = \text{map}\ (\text{Pair Discussion})\ wl\ )$
$\Delta_{\text{error}}\ (\sigma_1, sl_1, \sigma_2, sl_2)$	$sl_1 \neq [] \wedge$ $((\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid > \text{Reviewing}\wedge$ $\neg\ (\exists uid.\ \text{isRevNth}\ \sigma_1\ uid\ \text{PID N}))$ $\vee$ $(\exists cid.\ \text{PID} \in \text{paperIDs}\ \sigma_1\ cid \wedge \text{phase}\ \sigma_1\ cid > \text{Reviewing} \wedge$ $\text{fst}\ (\text{hd}\ sl_1) = \text{Reviewing}))$