Universal Composability is Robust Compilation

Marco Patrignani[1] Robert Künneman[2] Riad S. Wahby[3] Ethan Cecchetti[4]




January 2024 Toplas, journal first

1 UNITRENTO

2 CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

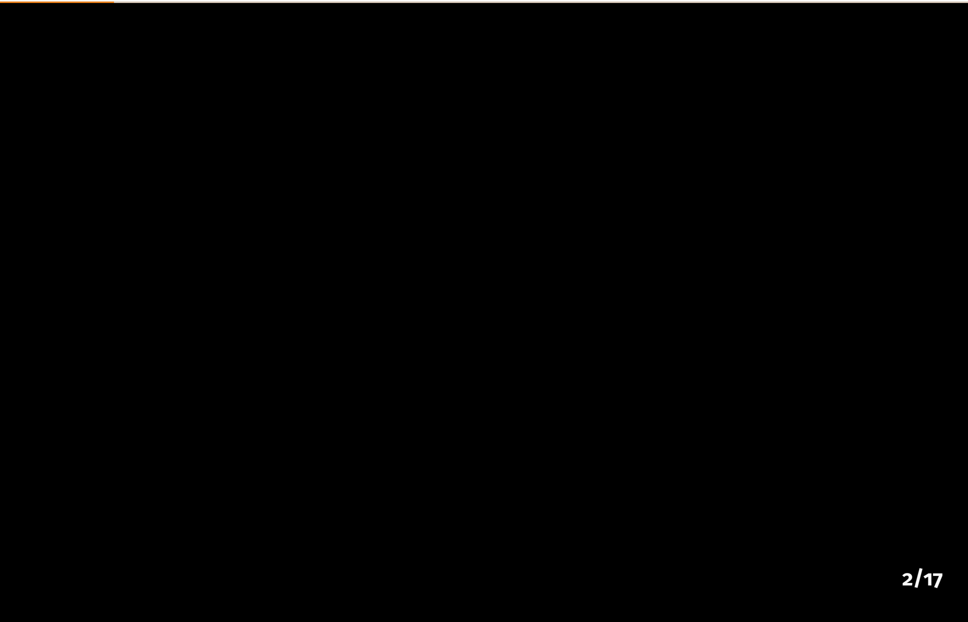
3 Carnegie Mellon University

4 WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

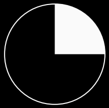
DISCLAIMER

pl != crypto

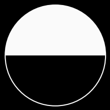
A Magic Trick



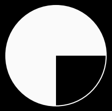
A Magic Trick



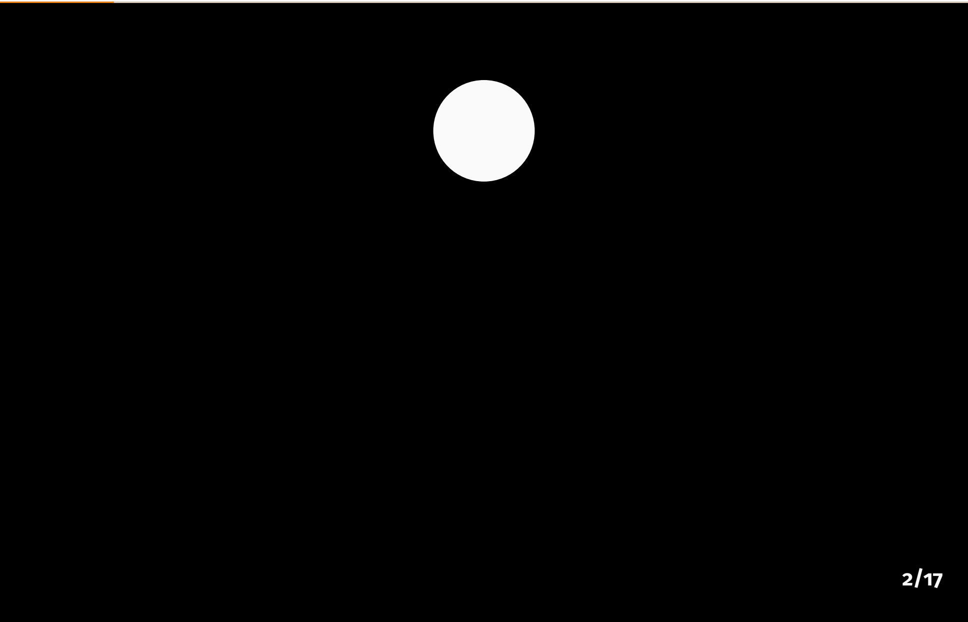
A Magic Trick



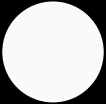
A Magic Trick



A Magic Trick

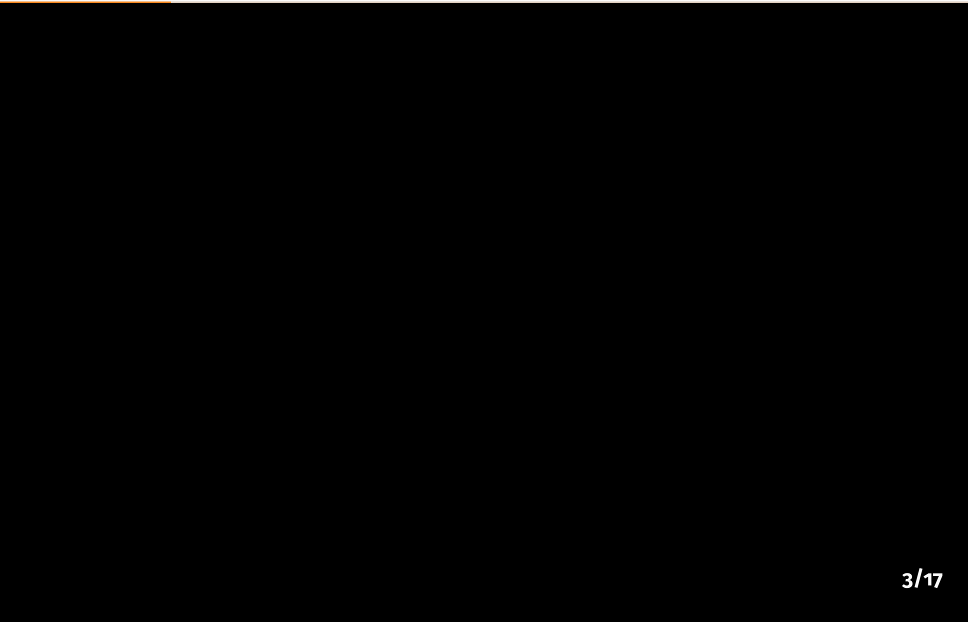


A Magic Trick



Spoiler: there are $0 + \epsilon$ hands raised

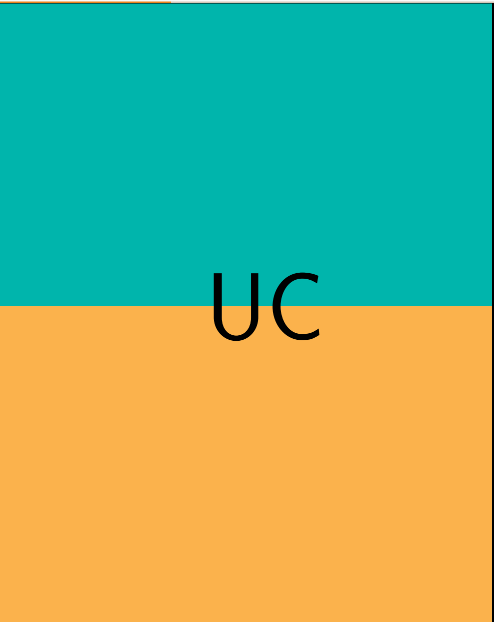
Motivation and the Journey



Motivation and the Journey

UC RC

Fields: UC



Universal Composability: UC

- gold standard for proving security of protocols under concurrent composition

Universal Composability: UC

- gold standard for proving security of protocols under concurrent composition
- overcomes security of protocol composition

Universal Composability: UC

- gold standard for proving security of protocols under concurrent composition
- overcomes security of protocol composition
- many flavours: UC[1], SaUCy [2], iUC [3], ...

[1] Canetti. 2001. “Universally composable security”

[2] Liao *et al.* 2019. “ILC: A Calculus for Composable, Computational Cryptography”

[3] Camenisch *et al.* 2019 “iUC: Flexible Universal Composability Made Simple”

Universal Composability: UC

- gold standard for proving security of protocols under concurrent composition
- overcomes security of protocol composition
- many flavours: UC[1], SaUCy [2], iUC [3], . . .

This work: axiomatic formalisation, geared towards the newer theories SaUCy and iUC

[1] Canetti. 2001. "Universally composable security"

[2] Liao *et al.* 2019. "ILC: A Calculus for Composable, Computational Cryptography"

[3] Camenisch *et al.* 2019 "iUC: Flexible Universal Composability Made Simple"

UC **Base Notions: ITMs** [4]

- protocols Π (using concrete crypto)

commitment for $b \in \{0,1\}$ *with SID sid:*

compute $G_{pk_b}(r)$ for random $r \in \{0,1\}^n$
set $y = G_{pk_b}(r)$ for $b = 0$, or $y = G_{pk_b}(r) \oplus \sigma$ for $b = 1$
send $(\texttt{Com}, sid, y)$ to the receiver

Upon receiving $(\texttt{Com}, sid, y)$ from P_i, P_j outputs $(\texttt{Receipt}, sid, cid, P_i, P_j)$

[4] From: Canetti, Fischlin. 2001. "Universally Composable Commitments"

UC **Base Notions: ITMs** [4]

- protocols Π (using concrete crypto)

commitment for $b \in \{0,1\}$ *with SID sid:*

compute $G_{pk_b}(r)$ for random $r \in \{0,1\}^n$
set $y = G_{pk_b}(r)$ for $b = 0$, or $y = G_{pk_b}(r) \oplus \sigma$ for $b = 1$
send $(\texttt{Com}, sid, y)$ to the receiver

Upon receiving $(\texttt{Com}, sid, y)$ from P_i, P_j outputs $(\texttt{Receipt}, sid, cid, P_i, P_j)$

- functionalities $\mathtt{F}$ (using abstract notions)

1. Upon receiving a value $(\texttt{Commit}, sid, P_i, P_j, b)$ from P_i, where $b \in \{0,1\}$, record the value b and send the message $(\texttt{Receipt}, sid, P_i, P_j)$ to P_j and $\mathcal{S}$. Ignore any subsequent $\texttt{Commit}$ messages.

[4] From: Canetti, Fischlin. 2001. "Universally Composable Commitments"

UC **Base Notions: ITMs** [4]

- protocols Π (using concrete crypto)

commitment for $b \in \{0,1\}$ *with SID sid:*

compute $G_{pk_b}(r)$ for random $r \in \{0,1\}^n$
set $y = G_{pk_b}(r)$ for $b = 0$, or $y = G_{pk_b}(r) \oplus \sigma$ for $b = 1$
send $(\texttt{Com}, sid, y)$ to the receiver

Upon receiving $(\texttt{Com}, sid, y)$ from P_i, P_j outputs $(\texttt{Receipt}, sid, cid, P_i, P_j)$

- functionalities $\mathtt{F}$ (using abstract notions)

1. Upon receiving a value $(\texttt{Commit}, sid, P_i, P_j, b)$ from P_i, where $b \in \{0,1\}$, record the value b and send the message $(\texttt{Receipt}, sid, P_i, P_j)$ to P_j and $\mathcal{S}$. Ignore any subsequent Commit messages.

- attackers $\mathtt{A}$ & $\mathtt{S}$ (corrupting parties etc.)

[4] From: Canetti, Fischlin. 2001. "Universally Composable Commitments"

UC **Base Notions: ITMs** [4]

- protocols Π (using concrete crypto)

commitment for $b \in \{0,1\}$ *with SID sid:*

compute $G_{pk_b}(r)$ for random $r \in \{0,1\}^n$
set $y = G_{pk_b}(r)$ for $b = 0$, or $y = G_{pk_b}(r) \oplus \sigma$ for $b = 1$
send $(\mathtt{Com}, sid, y)$ to the receiver

Upon receiving $(\mathtt{Com}, sid, y)$ from P_i, P_j outputs $(\mathtt{Receipt}, sid, cid, P_i, P_j)$

- functionalities $\mathcal{F}$ (using abstract notions)

1. Upon receiving a value $(\mathtt{Commit}, sid, P_i, P_j, b)$ from P_i, where $b \in \{0,1\}$, record the value b and send the message $(\mathtt{Receipt}, sid, P_i, P_j)$ to P_j and $\mathcal{S}$. Ignore any subsequent Commit messages.

- attackers $\mathcal{A}$ & $\mathcal{S}$ (corrupting parties etc.)
- environments $\mathcal{Z}$ (objective witness)

[4] From: Canetti, Fischlin. 2001. “Universally Composable Commitments”

Perfect (!!) UC

$$
\begin{tikzcd}
 & 0/1 & & & & 0/1 & \\
 & Z \arrow[u, Rightarrow] \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & & \approx & & Z \arrow[u, Rightarrow] \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\Pi \arrow[rr, leftrightarrow] & & A & & \mathtt{F} \arrow[rr, leftrightarrow] & & \mathtt{S}
\end{tikzcd}
$$

$\leftrightarrow$ represent communication channels

Perfect (!!) UC

$$
\begin{tikzcd}
 & 0/1 & \\
 & Z \arrow[u, Rightarrow] \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\Pi \arrow[rr, leftrightarrow] & & A
\end{tikzcd}
\quad \approx \quad
\begin{tikzcd}
 & 0/1 & \\
 & Z \arrow[u, Rightarrow] \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\mathtt{F} \arrow[rr, leftrightarrow] & & \mathtt{S}
\end{tikzcd}
$$

$\leftrightarrow$ represent communication channels

$$\Pi \vdash_{\mathrm{UC}} \mathtt{F} \stackrel{\text{def}}{=} \forall \text{poly } A, \exists \mathtt{S}, \forall Z.$$

$$\textsc{Exec}[Z, A, \Pi] \approx \textsc{Exec}[Z, \mathtt{S}, \mathtt{F}]$$

Perfect (!!) UC (computational UC in Künneman et al. CSF'24)

$$
\begin{tikzcd}
 & 0/1 & \\
 & Z \arrow[u, Rightarrow] \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\Pi \arrow[rr, leftrightarrow] & & A
\end{tikzcd}
\quad \approx \quad
\begin{tikzcd}
 & 0/1 & \\
 & Z \arrow[u, Rightarrow] \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\mathtt{F} \arrow[rr, leftrightarrow] & & \mathtt{S}
\end{tikzcd}
$$

$\leftrightarrow$ represent communication channels

$$\Pi \vdash_{UC} \mathtt{F} \stackrel{\text{def}}{=} \forall \text{poly } A, \exists \mathtt{S}, \forall Z.$$

$$\textsc{Exec}[Z, A, \Pi] \approx \textsc{Exec}[Z, \mathtt{S}, \mathtt{F}]$$

UC Beneifts: Compositionality

- if $\Pi_1 \vdash_{\mathsf{UC}} \mathtt{F}_1$
- and $\Pi_{\mathsf{big}} \stackrel{\mathsf{def}}{=} \Pi_{\mathsf{part}}[\Pi_1]$
- and $\mathtt{F}_{\mathtt{big}} \stackrel{\mathsf{def}}{=} \Pi_{\mathsf{part}}[\mathtt{F}_1]$

UC **Beneifts: Compositionality**

- if $\Pi_1 \vdash_{\mathrm{UC}} \mathtt{F}_1$
- and $\Pi_{\mathrm{big}} \stackrel{\mathrm{def}}{=} \Pi_{\mathrm{part}}[\Pi_1]$
- and $\mathtt{F}_{\mathtt{big}} \stackrel{\mathrm{def}}{=} \Pi_{\mathrm{part}}[\mathtt{F}_1]$

recall they are all ITMs

UC **Beneifts: Compositionality**

- if $\Pi_1 \vdash_{\text{UC}} \mathtt{F}_1$
- and $\Pi_{\text{big}} \stackrel{\text{def}}{=} \Pi_{\text{part}}[\Pi_1]$
- and $\mathtt{F}_{\mathtt{big}} \stackrel{\text{def}}{=} \Pi_{\text{part}}[\mathtt{F}_1]$
- then $\Pi_{\text{big}} \vdash_{\text{UC}} \mathtt{F}_{\mathtt{big}}$

UC **Beneifts: Compositionality**

- if $\Pi_1 \vdash_{\mathsf{UC}} \mathtt{F}_1$
- and $\Pi_{\mathsf{big}} \stackrel{\mathsf{def}}{=} \Pi_{\mathsf{part}}[\Pi_1]$
- and $\mathtt{F}_{\mathtt{big}} \stackrel{\mathsf{def}}{=} \Pi_{\mathsf{part}}[\mathtt{F}_1]$
- then $\Pi_{\mathsf{big}} \vdash_{\mathsf{UC}} \mathtt{F}_{\mathtt{big}} = \Pi_{\mathsf{part}}[\Pi_1] \vdash_{\mathsf{UC}} \mathtt{F}_{\mathtt{big}}$

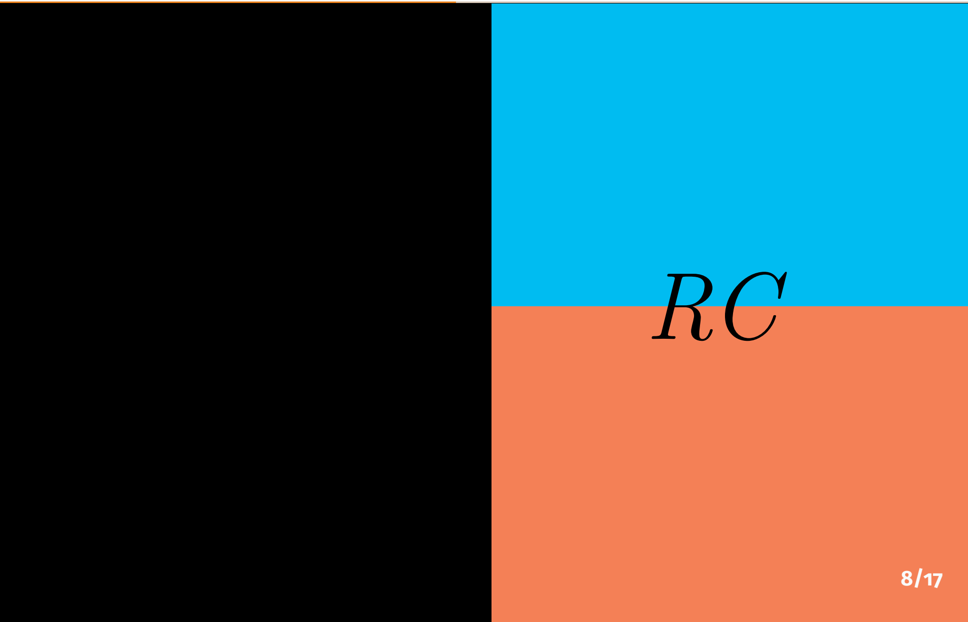
UC **Beneifts: Compositionality**

- if $\Pi_1 \vdash_{\mathsf{UC}} \mathtt{F}_1$
- and $\Pi_{\mathsf{big}} \stackrel{\mathsf{def}}{=} \Pi_{\mathsf{part}}[\Pi_1]$
- and $\mathtt{F}_{\mathtt{big}} \stackrel{\mathsf{def}}{=} \Pi_{\mathsf{part}}[\mathtt{F}_1]$
- then $\Pi_{\mathsf{big}} \vdash_{\mathsf{UC}} \mathtt{F}_{\mathtt{big}} =$
  $\Pi_{\mathsf{part}}[\Pi_1] \vdash_{\mathsf{UC}} \mathtt{F}_{\mathtt{big}} =$
  $\Pi_{\mathsf{part}}[\Pi_1] \vdash_{\mathsf{UC}} \Pi_{\mathsf{part}}[\mathtt{F}_1]$

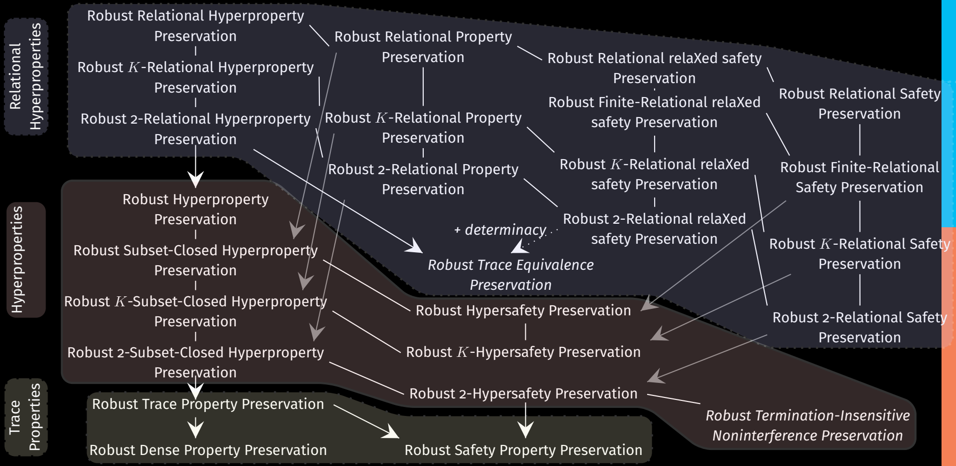
Fields

UC RC

Fields: RC

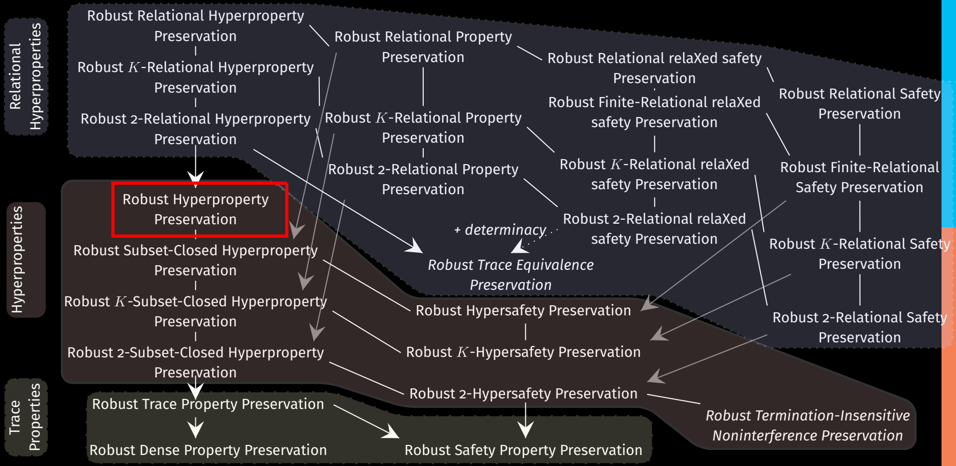


Robust Compilation [5]



[5] Abate *et al.* 2019. "Journey Beyond Full Abstraction ..."

Robust Compilation [5]



[5] Abate *et al.* 2019. "Journey Beyond Full Abstraction …"

Robust Hyperproperty Preservation: RHC

$$\llbracket P \rrbracket \overset{\bar{t}}{\underset{\rightsquigarrow}{\bowtie}} \mathbf{A} \iff P \overset{\bar{t}}{\underset{\rightsquigarrow}{\bowtie}} A$$

Robust Hyperproperty Preservation: RHC

$$\llbracket P \rrbracket \overset{\overline{t}}{\overset{\wr}{\bowtie}} \mathbf{A} \iff P \overset{\overline{t}}{\overset{\wr}{\bowtie}} A$$

$$\llbracket\cdot\rrbracket \vdash RHC \overset{\text{def}}{=} \forall P, \mathbf{A}.\ \exists A.\ \forall \overline{t}.$$
$$\mathbf{A} \bowtie \llbracket P \rrbracket \rightsquigarrow \overline{t} \iff A \bowtie P \rightsquigarrow \overline{t}$$

For any language S and $\mathbf{T}$

A Closer Look

$$\forall \text{poly } \mathsf{A}, \exists \mathsf{S}, \forall \mathsf{Z}.$$

$$\begin{tikzcd}[column sep=small]
 & 0/1 & & & & 0/1 & \\
 & \mathsf{Z} \arrow[u, Rightarrow] \arrow[dl, <->] \arrow[dr, <->] & & \approx & & \mathsf{Z} \arrow[u, Rightarrow] \arrow[dl, <->] \arrow[dr, <->] & \\
\Pi \arrow[rr, <->] & & \mathsf{A} & & \mathtt{F} \arrow[rr, <->] & & \mathtt{S}
\end{tikzcd}$$

$$\forall P, \mathbf{A}. \exists A. \forall \bar{t}.$$

$$\begin{tikzcd}
 & \bar{t} \arrow[d, squiggly, <-] & & & \bar{t} \arrow[d, squiggly, <-] & \\
\llbracket P \rrbracket & \bowtie & \mathbf{A} \arrow[r, Leftrightarrow] & P & \bowtie & A
\end{tikzcd}$$

Our (Isabelle'd) Connection

UC			RC
protocol	$\sqcap$	$[\![P]\!]$	compiled program
concrete attacker	A	$\mathbf{A}$	target context
ideal functionality	$\mathtt{F}$	P	source program
simulator	$\mathtt{S}$	A	source context
environment, output	Z, 0/1	$\overline{t}$, $\rightsquigarrow$	trace, semantics
communication	$\leftrightarrow$	$\bowtie$	linking
probabilistic equiv.	$\approx$	$\iff$	trace equality

Our (Isabelle'd) Connection

UC			RC
protocol	Π	$[\![P]\!]$	compiled program
concrete attacker	A	$\mathbf{A}$	target context
ideal functionality	$\mathtt{F}$	P	source program
simulator	$\mathtt{S}$	A	source context
environment, output	Z, 0/1	$\overline{t}$, $\rightsquigarrow$	trace, semantics
communication	$\leftrightarrow$	$\bowtie$	linking
probabilistic equiv.	$\approx$	$\Longleftrightarrow$	trace equality
human translation $\Pi \to \mathtt{F}$		$[\![\cdot]\!]$: $P \to \mathbf{P}$ compiler	

Why Should You Care?

Prove RHC via UC

(e.g., Viaduct …Acay et al PLDI'21)

Why Should You Care?

Prove RHC via UC

(e.g., Viaduct …Acay et al PLDI'21)

Admittedly, less explored,

(is there more?)

Why Should You Care?

Why Should You Care?

Mechanise UC proofs with program analysis tools

(Deepsec, Cryptoverif, Squirrel, etc)

Why Should You Care?

Mechanise UC proofs with program analysis tools

(Deepsec, Cryptoverif, Squirrel, etc)

as in computer-aided crypto

How? The 1-bit Commitment Example

- Write protocol and `functionality` as Deepsec processes

How? The 1-bit Commitment Example

- Write protocol and `functionality` as Deepsec processes
- Start building the missing `ideal processes` (90%) using:

How? The 1-bit Commitment Example

- Write protocol and `functionality` as Deepsec processes
- Start building the missing `ideal processes` (90%) using:
  - backtranslation (from secure compilation)
  - and dummy attacker theorem (from this work)

How? The 1-bit Commitment Example

- Write protocol and `functionality` as Deepsec processes
- Start building the missing `ideal processes` (90%) using:
  - backtranslation (from secure compilation)
  - and dummy attacker theorem (from this work)
- Fill the `missing lines` (4!)

How? The 1-bit Commitment Example

- Write protocol and `functionality` as Deepsec processes
- Start building the missing `ideal processes` (90%) using:
  - backtranslation (from secure compilation)
  - and dummy attacker theorem (from this work)
- Fill the `missing lines` (4!)
- Wrap real and `ideal` processes with an environment proxy to regulate scheduling

How? The 1-bit Commitment Example

- Write protocol and `functionality` as Deepsec processes
- Start building the missing `ideal processes` (90%) using:
  - backtranslation (from secure compilation)
  - and dummy attacker theorem (from this work)
- Fill the `missing lines` (4!)
- Wrap real and `ideal` processes with an environment proxy to regulate scheduling
- Add the missing `lines` for adaptive corruption (binding or hiding, not both)

What Will You Find in the (64 pp) Paper?

1. Axiomatised UC semantics

What Will You Find in the (64 pp) Paper?

1. Axiomatised UC semantics
2. Isabelle'd the connection

What Will You Find in the (64 pp) Paper?

1. Axiomatised UC semantics
2. Isabelle'd the connection
3. Formalised conditions to use the connection with any language

What Will You Find in the (64 pp) Paper?

1. Axiomatised UC semantics
2. Isabelle'd the connection
3. Formalised conditions to use the connection with any language
4. Formalised composition axioms

What Will You Find in the (64 pp) Paper?

1. Axiomatised UC semantics
2. Isabelle'd the connection
3. Formalised conditions to use the connection with any language
4. Formalised composition axioms
5. Mechanised UC proof for 1-bit commitment for static & adaptive corruption

What Will You Find in the (64 pp) Paper?

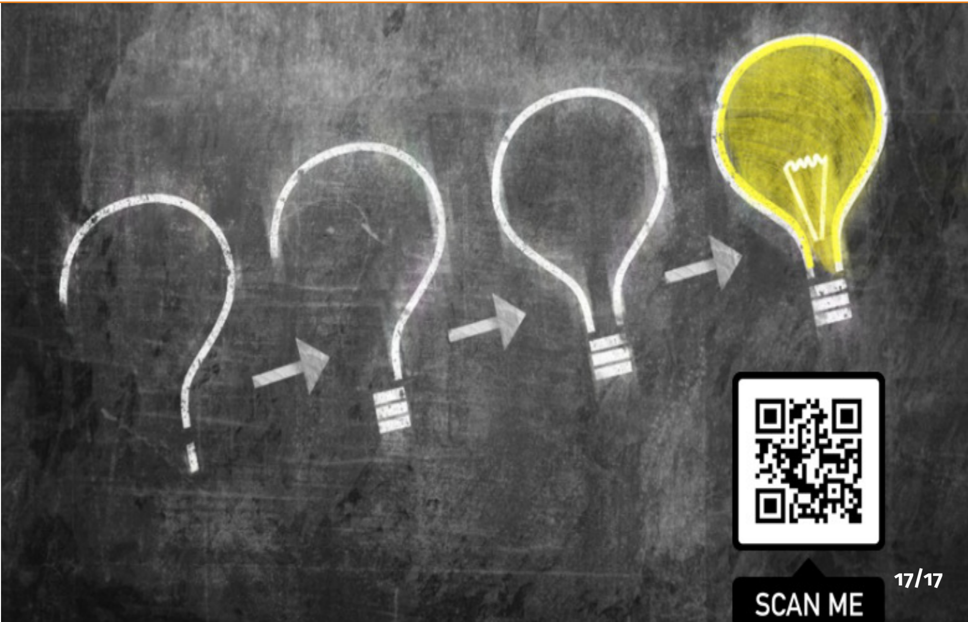
1. Axiomatised UC semantics
2. Isabelle'd the connection
3. Formalised conditions to use the connection with any language
4. Formalised composition axioms
5. Mechanised UC proof for 1-bit commitment for static & adaptive corruption
6. A lot of insights

What Will You Find in the (64 pp) Paper?

A security proof between ITMs is a UC proof,
a security proof between arbitrary languages is a RC one.

5. Mechanised UC proof for 1-bit commitment for static & adaptive corruption
6. A lot of insights

Questions?



The Full Abstraction (false) Conjecture

$$
\begin{tikzcd}
 & 0/1 & \\
 & Z \arrow[u, Rightarrow] \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\Pi \arrow[rr, leftrightarrow] & & A
\end{tikzcd}
\approx
\begin{tikzcd}
 & 0/1 & \\
 & Z \arrow[u, Rightarrow] \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\mathtt{F} \arrow[rr, leftrightarrow] & & \mathtt{S}
\end{tikzcd}
$$

$$
\begin{array}{c}
[\![P_1]\!] \bowtie \mathbf{A} \Downarrow \\
\Updownarrow \\
[\![P_2]\!] \bowtie \mathbf{A} \Downarrow
\end{array}
\iff
\begin{array}{c}
P_1 \bowtie A \Downarrow \\
\Updownarrow \\
P_2 \bowtie A \Downarrow
\end{array}
$$

FAC is relational, RHC is propositional, like UC

What is the Compiler?

- seemingly-degenerate (translate one concrete input to one concrete output)

What is the Compiler?

- seemingly-degenerate (translate one concrete input to one concrete output)
- the connection works with any compiler!

What is the Compiler?

- seemingly-degenerate (translate one concrete input to one concrete output)
- the connection works with any compiler!
- if only there were a protocol definition language …(future work)

Composition Operators

- Linking
- Program FFI
- Attacker FFI
- Complete FFI

Composition Operators

- Linking
- Program FFI
- Attacker FFI
- Complete FFI

- Just program-level linking in any PL

Composition Operators

- Linking
- Program FFI
- Attacker FFI
- Complete FFI
- Just program-level linking in any PL
- Must follow 3 (obvious) axioms

The Dummy Attacker

- In UC, replace A with a dummy proxy

The Dummy Attacker

- In UC, replace A with a dummy proxy
- 4 (obvious) Axioms provide the same theorem in RC

The Dummy Attacker

- In UC, replace A with a dummy proxy
- 4 (obvious) Axioms provide the same theorem in RC
  thus, no need to do induction, just reason about the $source + simulator$ and $\mathbf{target}$ programs (with tools)