Universal Composability is Secure Compilation

Marco Patrignani[1,2] Riad S. Wahby[1] Robert Künneman[2]

25th January 2020

1 Stanford University

2 CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Goal

Unveil a similarity between two fields

Goal

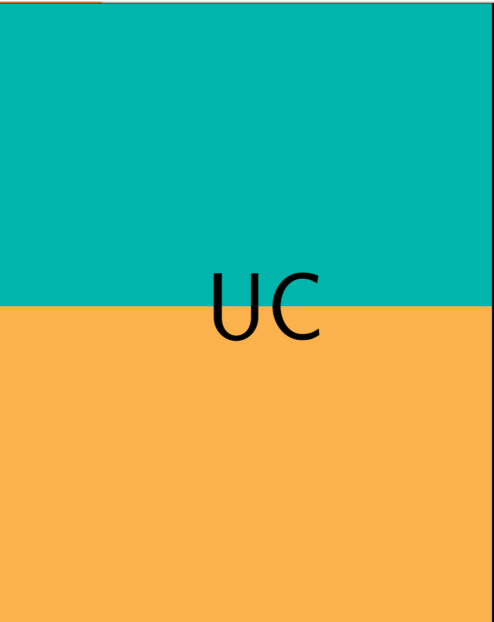
Unveil a similarity between two fields

Explore how each field can benefit from the other

Fields

UC SC

Fields: UC



Universal Composability: UC

- gold standard for proving security of crypto protocols under concurrent composition

Universal Composability: UC

- gold standard for proving security of crypto protocols under concurrent composition
- overcome main drawback in protocol vulnerabilities: composition

Universal Composability: UC

- gold standard for proving security of crypto protocols under concurrent composition
- overcome main drawback in protocol vulnerabilities: 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 crypto protocols under concurrent composition
- overcome main drawback in protocol vulnerabilities: composition
- many flavours: UC[1], SaUCy [2], iUC [3]

This talk: generic presentation, 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 $\mathcal{F}$ (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.

[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 $(\texttt{Com}, sid, y)$ to the receiver

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

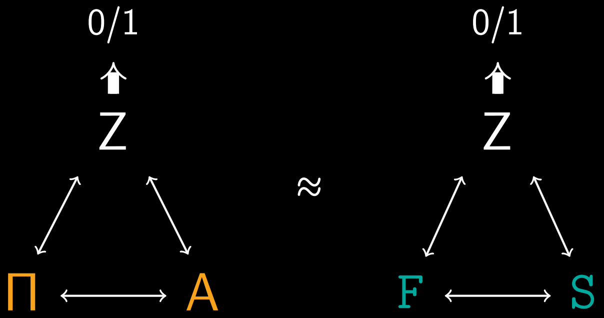
- functionalities F (using abstract notions)

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 A & S (corrupting parties etc.)
- environments Z (objective witness)

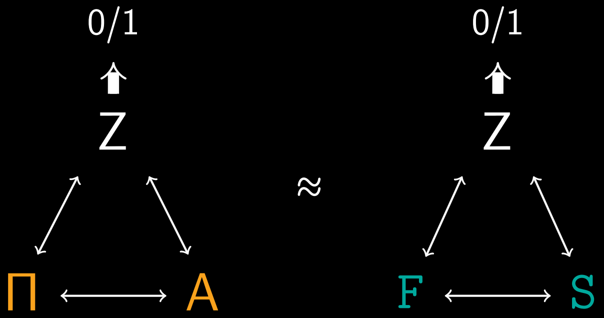
[4] From: Canetti, Fischlin. 2001. "Universally Composable Commitments"

UC (Semi-formally)



$\leftrightarrow$ represent communication channels

UC (Semi-formally)



$\leftrightarrow$ represent communication channels

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

UC, **Pros and Cons**

- modularise protocols
- small building blocks
- reusable results

UC, **Pros and Cons**

- modularise protocols
- small building blocks
- reusable results

1. informal formalism
2. pseudocode protocols
3. (PL-wise) informal proofs
4. no (ish) mechanisation

UC, **Pros and Cons**

- modularise protocols
- small building blocks
- reusable results

1. informal formalism
2. pseudocode protocols
3. (PL-wise) informal proofs
4. no (ish) mechanisation

Existing work (SaUCy and iUC): points 1 and 2

UC, **Pros and Cons**

- modularise protocols
- small building blocks
- reusable results

1. informal formalism
2. pseudocode protocols
3. (PL-wise) informal proofs
4. no (ish) mechanisation

Existing work (SaUCy and iUC): points 1 and 2

Our work: points 3 and 4

UC **Beneifts: Compositionality**

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

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 $\mathtt{F}_{\mathtt{big}} \stackrel{\text{def}}{=} \Pi_{\text{part}}[\mathtt{F}_1]$

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

UC **Beneifts: Compositionality**

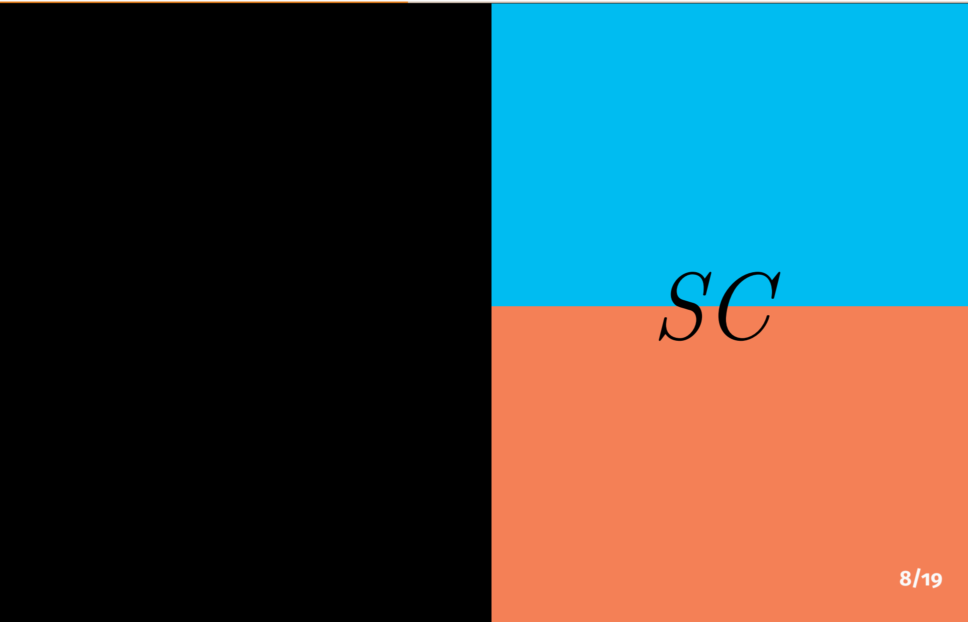
- if $\Pi_1 \vdash_{\mathrm{UC}} \mathtt{F}_1$
- and $\Pi_{\mathrm{big}} \overset{\mathrm{def}}{=} \Pi_{\mathrm{part}}[\Pi_1]$
- and $\mathtt{F}_{\mathtt{big}} \overset{\mathrm{def}}{=} \Pi_{\mathrm{part}}[\mathtt{F}_1]$
- then $\Pi_{\mathrm{big}} \vdash_{\mathrm{UC}} \mathtt{F}_{\mathtt{big}} =$
  $\Pi_{\mathrm{part}}[\Pi_1]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

Fields

UC

SC

Fields: SC



Secure Compilation: SC

Secure Compilation: SC

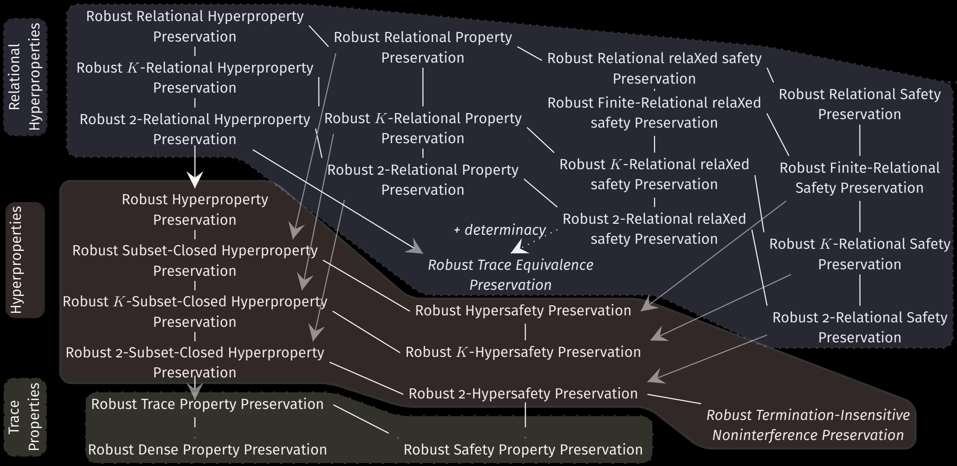
- many criteria: FAC[5], TPC[6], $RSCC$[7], ...

[5] Abadi. 1998. "Protection in Programming-Language Translations"

[6] Patrignani, Garg. 2017. "Secure Compilation and Hyperproperties Preservation"

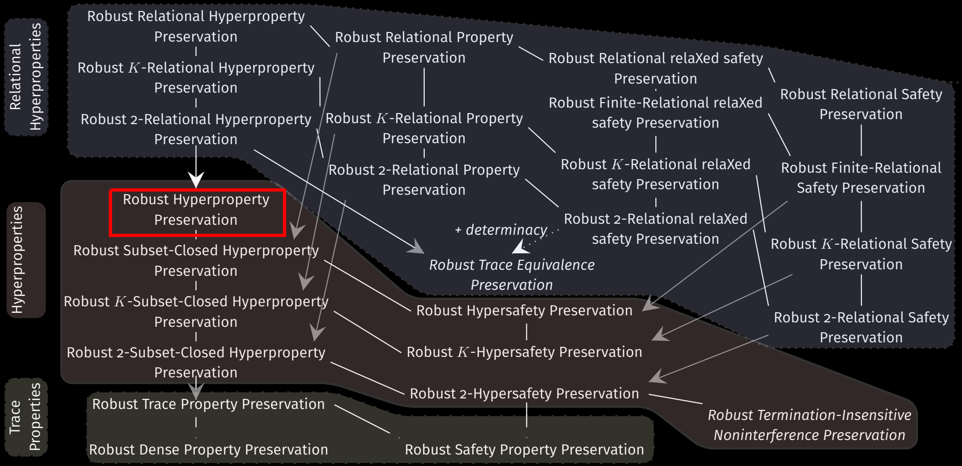
[7] Abate *et al.* 2018. "When Good Components Go Bad ..."

Robust Criteria for SC



Abate *et al.* 2019. “Journey Beyond Full Abstraction …”

Robust Criteria for SC



Abate *et al.* 2019. “Journey Beyond Full Abstraction …”

Robust Hyperproperty Preservation: RHC

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

Robust Hyperproperty Preservation: RHC

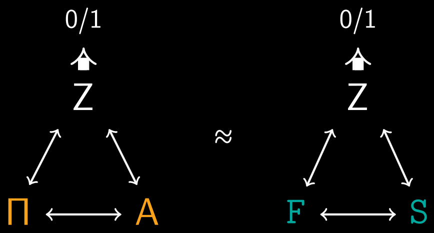
$$
\begin{array}{ccccccc}
 & \overline{t} & & & & \overline{t} & \\
 & \rightsquigarrow & & & & \rightsquigarrow & \\
[\![P]\!] & \bowtie & \mathbf{A} & \iff & P & \bowtie & A
\end{array}
$$

$$
[\![\cdot]\!] \vdash RHC \overset{\text{def}}{=} \forall P, \mathbf{A}. \exists A. \forall \overline{t}.
$$

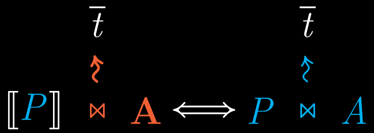
$$
\mathbf{A} \bowtie [\![P]\!] \rightsquigarrow \overline{t} \iff A \bowtie P \rightsquigarrow \overline{t}
$$

A Closer Look

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



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



Analogy

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

Analogy

UC		SC	
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 Π $\rightarrow$ $\mathtt{F}$		$[\![\cdot]\!]$: $P \rightarrow \mathbf{P}$ compiler	
general composition result		...	

Our Claim

UC and RHC are similar enough so that we can reuse metatheoretical results of one system for the other

Benefits

Cryptographers:

- must specify hidden UC assumptions[8]
- more formal UC proofs
- mechanisation of UC results

[8] As advocated by: Barbosa *et al.* 2019. "SoK: Computer-aided Cryptography"

Benefits

Cryptographers:

- must specify hidden UC assumptions[8]
- more formal UC proofs
- mechanisation of UC results

Secure-compilationers:

- understand composition of SC results

[8] As advocated by: Barbosa *et al.* 2019. “SoK: Computer-aided Cryptography”

Benefits

Cryptographers:

- must specify hidden UC assumptions[8]
- more formal UC proofs
- mechanisation of UC results

Secure-compilationers:

- understand composition of SC results

more?

[8] As advocated by: Barbosa *et al.* 2019. “SoK: Computer-aided Cryptography”

UC **Roadmap**

1.
   - formalise simple functionalities and protocols in ILC
   - prove their compiler is RHC

UC **Roadmap**

1.
- formalise simple functionalities and protocols in ILC
- prove their compiler is RHC

2.
- formally prove (a version of) UC (iUC) and RHC are equivalent

SC Roadmap

- RHC defined for $[\![\cdot]\!]$ but paper mentions chains = compiler, linker(s), … = $([\![\cdot]\!], \bowtie, \bowtie)$

SC Roadmap

- RHC defined for $[\![\cdot]\!]$ but paper mentions chains = compiler, linker(s), … = $([\![\cdot]\!], \bowtie, \bowtie)$

Assuming these are RHC:

- $([\![\cdot]\!]^{S}_{\mathbf{T}}, \bowtie, \bowtie)$ $\quad$ $([\![\cdot]\!]^{O}_{\mathbf{T}}, \bowtie, \bowtie)$ $\quad$ $([\![\cdot]\!]^{\mathbf{T}}_{\mathbf{B}}, \bowtie, \bowtie)$

What can we say about:

- $([\![\cdot]\!]^{S}_{\mathbf{B}} = [\![\cdot]\!]^{S}_{\mathbf{T}} \circ [\![\cdot]\!]^{\mathbf{T}}_{\mathbf{B}}, \bowtie, \bowtie)$?

SC Roadmap

- RHC defined for $[\![\cdot]\!]$ but paper mentions chains = compiler, linker(s), … = $([\![\cdot]\!], \bowtie, \bowtie)$

Assuming these are RHC:

- $([\![\cdot]\!]^{S}_{\mathbf{T}}, \bowtie, \bowtie)$ $\quad ([\![\cdot]\!]^{O}_{\mathbf{T}}, \bowtie, \bowtie)$ $\quad ([\![\cdot]\!]^{\mathbf{T}}_{\mathbf{B}}, \bowtie, \bowtie)$

What can we say about:

- $([\![\cdot]\!]^{S}_{\mathbf{B}} = [\![\cdot]\!]^{S}_{\mathbf{T}} \circ [\![\cdot]\!]^{\mathbf{T}}_{\mathbf{B}}, \bowtie, \bowtie)$?
- $([\![\cdot]\!]^{S \cup O}_{\mathbf{T}} = [\![\cdot]\!]^{S}_{\mathbf{T}} \cup [\![\cdot]\!]^{O}_{\mathbf{T}}, \bowtie \cup \bowtie, \bowtie)$?

SC Roadmap

- RHC defined for $[\![\cdot]\!]$ but paper mentions chains = compiler, linker(s), … = $([\![\cdot]\!], \bowtie, \bowtie)$

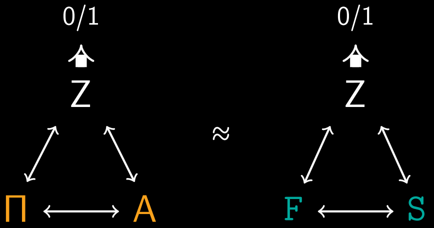
Assuming these are RHC:

- $([\![\cdot]\!]^{S}_{\mathbf{T}}, \bowtie, \bowtie)$ $([\![\cdot]\!]^{O}_{\mathbf{T}}, \bowtie, \bowtie)$ $([\![\cdot]\!]^{\mathbf{T}}_{\mathbf{B}}, \bowtie, \bowtie)$

What can we say about:

- $([\![\cdot]\!]^{S}_{\mathbf{B}} = [\![\cdot]\!]^{S}_{\mathbf{T}} \circ [\![\cdot]\!]^{\mathbf{T}}_{\mathbf{B}}, \bowtie, \bowtie)$?
- $([\![\cdot]\!]^{S \cup O}_{\mathbf{T}} = [\![\cdot]\!]^{S}_{\mathbf{T}} \cup [\![\cdot]\!]^{O}_{\mathbf{T}}, \bowtie \cup \bowtie, \bowtie)$?
- $\mathbf{P} = [\![P]\!]^{S}_{\mathbf{T}} \bowtie [\![P]\!]^{O}_{\mathbf{T}}$

But Fully Abstract Compilation …



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

FAC is relational, RHC is propositional, like UC

But Fully Abstract Compilation …

Questions?

But What is the $\forall P$?

- each pair P-$[\![P]\!]$ is a pair of UC $\mathtt{F}$-$\sqcap$
- $[\![P]\!]^{S}_{\mathbf{T}} = \begin{cases} \mathbf{P} & \text{if } \mathbf{P} \vdash_{\mathsf{UC}} P \\ P & \text{otherwise} \end{cases}$

  in this interpretation, S and $\mathbf{T}$ are ITMs

But Attackers and Environments ...

- UC works employ a dummy attacker
- the $\forall Z$ accounts for attacker behaviour
- Z has some “objective” behaviour

But Attackers and Environments …

- UC works employ a dummy attacker
- the $\forall Z$ accounts for attacker behaviour
- Z has some “objective” behaviour
- we leave the attacker business in $\mathbf{A}$
- and the semantics ($\rightsquigarrow$) to the objectivity

this is similar to the EasyUC work

But UC **was Mechanised in EasyCrypt**[9]

[9] Canetti *et al.* 2019. “EasyUC: Using EasyCrypt to Mechanize Proofs of Universally Composable Security”

But UC was Mechanised in EasyCrypt[9]

- with a titanic effort

[9] Canetti *et al.* 2019. "EasyUC: Using EasyCrypt to Mechanize Proofs of Universally Composable Security"

But UC **was Mechanised in EasyCrypt**[9]

- with a titanic effort
- our analogy is tool-indipendent

[9] Canetti *et al.* 2019. “EasyUC: Using EasyCrypt to Mechanize Proofs of Universally Composable Security”

But UC was Mechanised in EasyCrypt[9]

- with a titanic effort
- our analogy is tool-indipendent
- some similarities between the approaches (see next)

[9] Canetti *et al.* 2019. "EasyUC: Using EasyCrypt to Mechanize Proofs of Universally Composable Security"