Secure Compilation as Hyperproperties Preservation

**The problem**

What is a secure compiler?

I don’t know

what do you mean by secure?

**A Declassification Example**

Language $S$ has Bools

Language $T$ has Nats

$[.] : S \rightarrow T$

$\lambda x. \text{if interactions < 10 then } x \text{ else secret}$

$\lambda x. \text{if } x > 1 \text{ then secret else secret}$

$[.]$ is a Compiler from $S$ to $T$

Is $[.]$ secure?

no (but it is FAC!!)

**A 2nd Example**

What if the compiler produces this:

$\lambda x. \text{if } x > 1 \text{ then fail else secret}$

elif interactions <10 then $x$ else secret

Is $[.]$ secure now?

yes

**Hyperproperties**

Clarkson and Schneider ‘10

formalise any program property

Interesting subclass: *hypersafety*: “something bad does not happen”

How does this relate to compilation?

First result

TPC preserves *hypersafety*!

If a $S$ program has some hypersafety then its compilation also has it.

We know the precise security implications of TPC!

What about related work?

like fully-abstract compilers

Second result

Most works in compiler security prove compiler full abstraction FAC

Abadi ‘00

We know the precise security implications of FAC!

**The Definition**

$T$ race

$P$ reserving

$C$ compiler

- on “good” inputs, act “like” $S$

so TPC implies compiler correctness!

- on “bad” ones, act *opaquely* *(fail)*

$fail$ must not be expressible in $S$

Why is this secure?