Expressive Authorization Policies using Computation Principals

Anitha Gollamudi and Stephen Chong
Harvard University

Goal

Express Authorization Policies involving **Computations**

CODE

CODE

CODE

Pay $42
to IRS

Pay $42
to IRS

Pay $42
to IRS

CODE

speaks for

CODE

is a

Computation Principal

CODE

Examples

Examples

Hash Digest

Mobile code

Smart Contracts

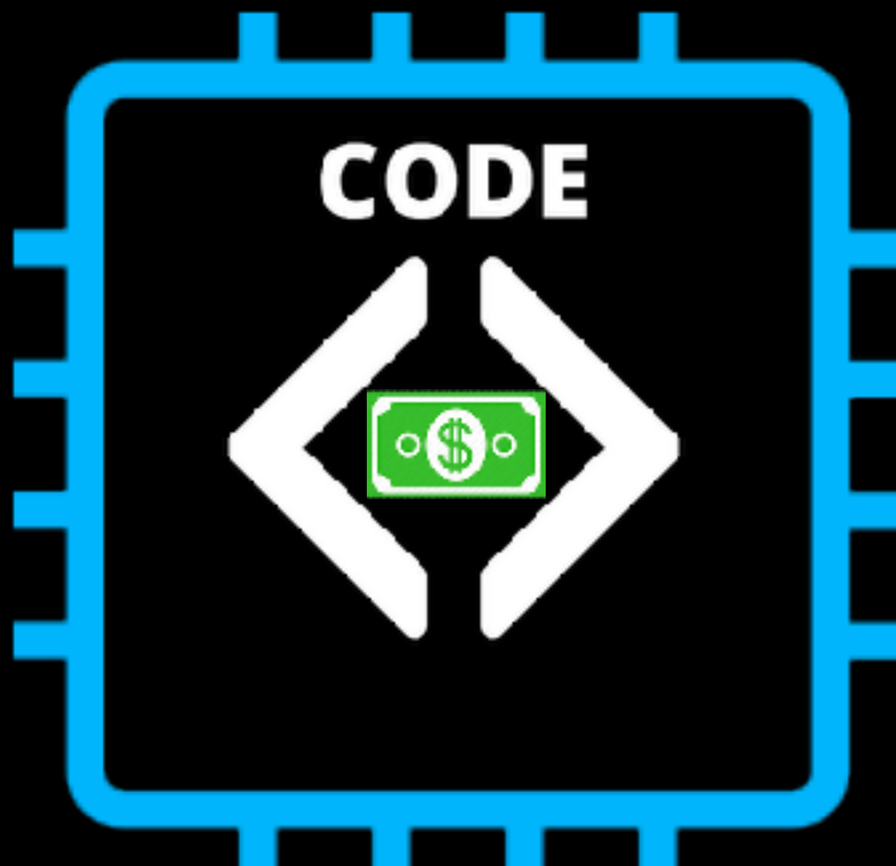
Trusted Execution Environments

Trusted Execution Environment (TEE)

- TEE provides confidentiality and integrity to the code running inside
  - e.g. Intel SGX, ARM TrustZone
- identity of TEE = identity of the code
- Offers remote code attestation

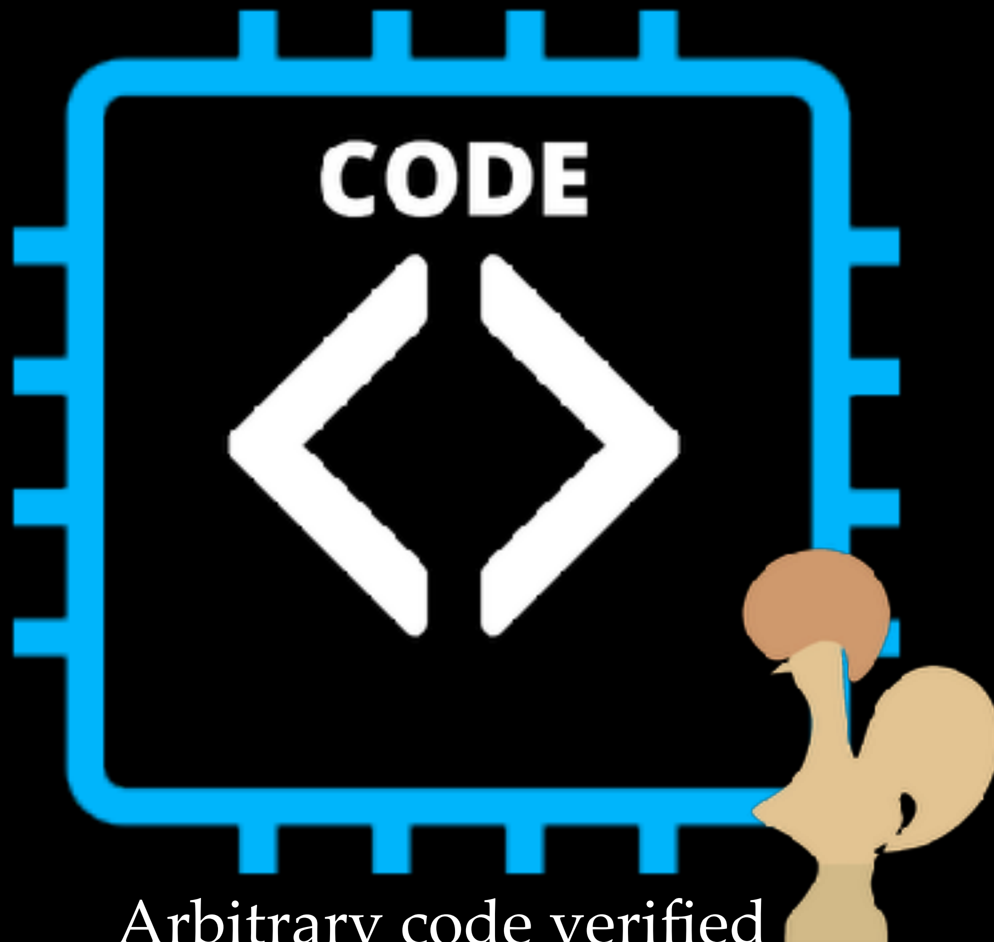
TEE enables delegating trust to a computation independent of the machine on which it is executed

Simple Policy



speaks for

Complex Policy



Arbitrary code verified by a proof assistant

speaks for

Contribution

An authorization logic with a mechanism that elegantly expresses trust directly to the code using ***Computation Principals***

Our authorization logic extends Abadi's Polymorphic Dependency Core Calculus (DCC)

Polymorphic DCC

- A calculus and authorization logic for access control
- Extends Moggi's computational lambda calculus
  - $(\eta_A\ e)$ protects e at level A
    - has the type "A says …"
  - bind x = $(\eta_A\ e)$ in e' privileged operation authorized by A
  - Polymorphism

Encoding “A speaks for B”

Encoding “A speaks for B”

Type (Proposition)	
$\forall X.\ A \text{ says } X \rightarrow B \text{ says } X$	

Encoding "A speaks for B"

Type (Proposition)	Term (Proof)
$\forall X.\ A \text{ says } X \rightarrow B \text{ says } X$	$\Lambda X.\ \lambda x{:}\ A \text{ says } X.$ $\text{bind } y = x \text{ in } (\eta_B\ y)$

Curry-Howard Correspondence

C-DCC

$e ::= \ldots \mid \boldsymbol{\mu}t.e \mid \mathrm{exec}(e, e')$

$\boldsymbol{\tau} ::= \ldots \mid \mathrm{CP}\ \boldsymbol{\mu}t.e$

C-DCC

Computation Expression

e ::= … | $\mathbf{\mu}$t.e | exec(e, e')

$\boldsymbol{\tau}$ ::= … | CP $\mathbf{\mu}$t.e

C-DCC

Computation Expression

$e ::= \ldots \mid \mu t.e \mid exec(e, e')$

$\tau ::= \ldots \mid CP\ \mu t.e$

Computation Principal

C-DCC

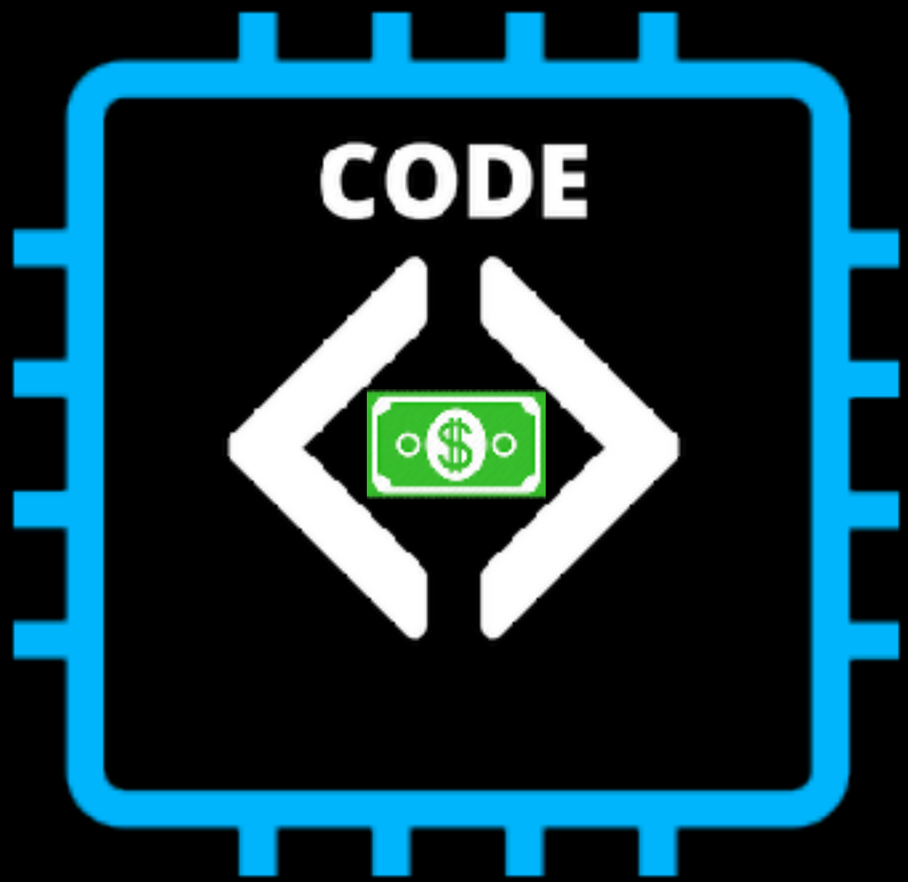
Computation Expression

$e ::= \ldots \mid \boldsymbol{\mu}t.e \mid exec(e, e')$

$\boldsymbol{\tau} ::= \ldots \mid CP\ \boldsymbol{\mu}t.e$

Computation Principal

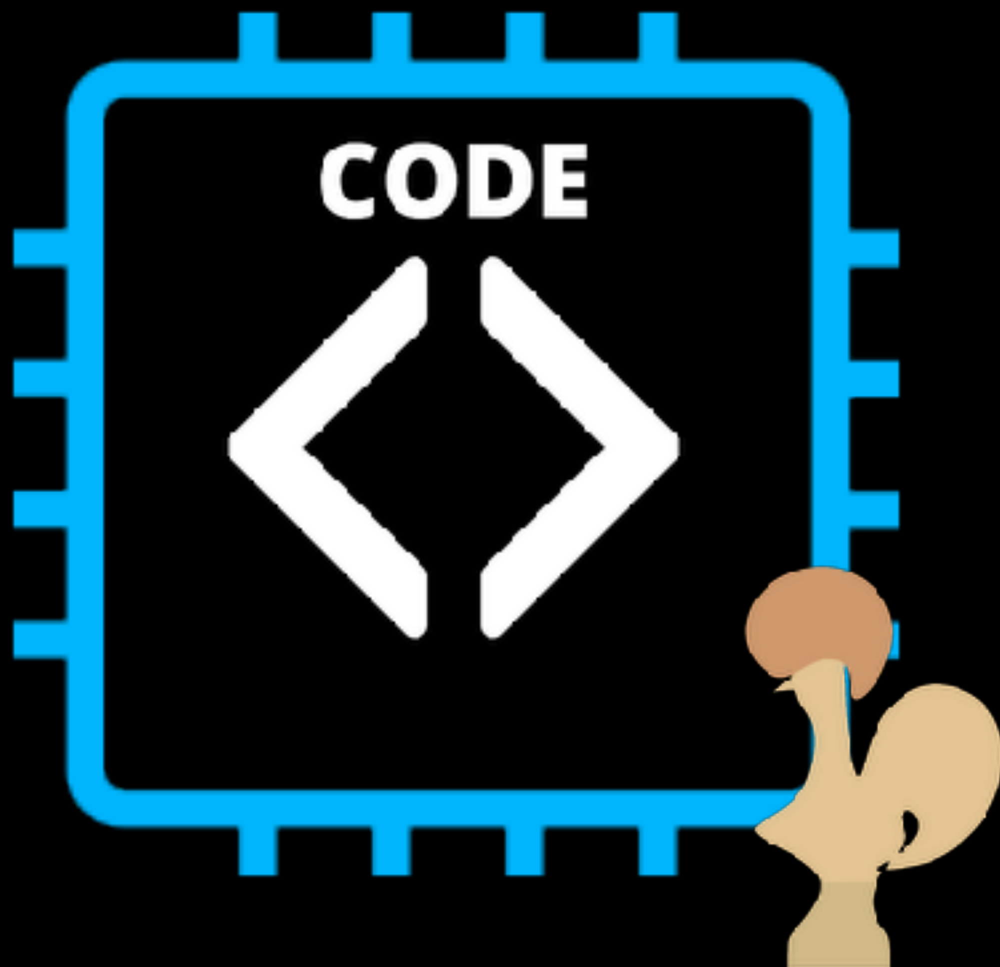
Run computation



speaks for

∀X. (CP μt. <  >) says X → A says X

Dependent Type



speaks for

$$\forall P{::}\text{Computation Principals. (Verified } P) \rightarrow$$
$$\forall X.\ \mathbf{P} \text{ says } X \rightarrow \mathbf{A} \text{ says } X$$

Predicates on Computation Principals