Qualification Exam

A Domain-Specific Programming Language for Secure Multiparty Computation

Janus Dam Nielsen

Thesis

Creating tools with strong security guaranties which exploits the benefits obtained by combining confidential information without compromising it, is feasible and useful.

Report on SMCL

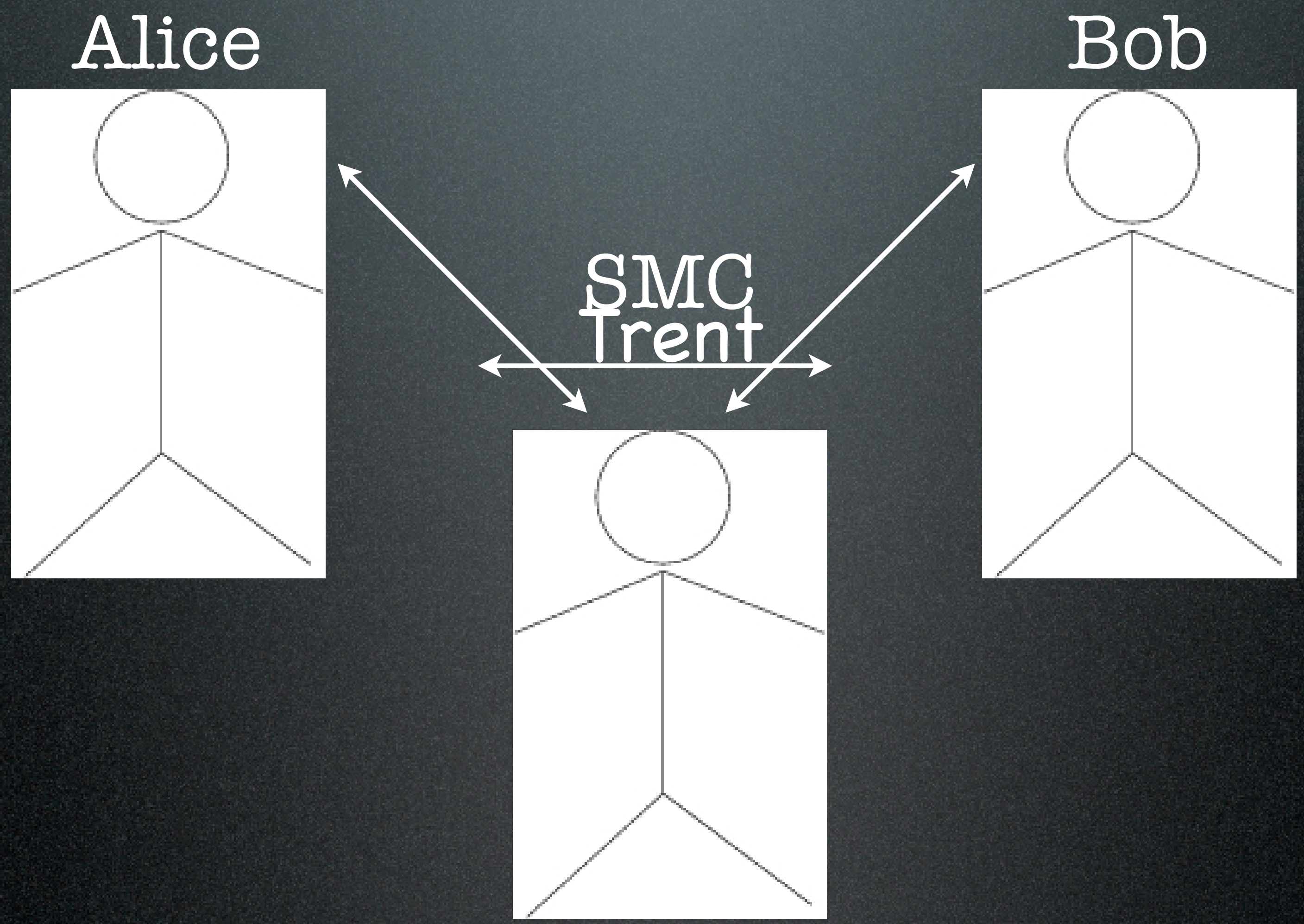
Overview

- Secure Multiparty Computation
- SMCL Concepts
- An example
- Security - what, why
- Efficiency
- Future Work
- Conclusion

Secure Multiparty Computation

- n parties P1,...,Pn wish to jointly compute the computable function: f(x1,...,xn)
- Party Pi only knows the input value xi which must be kept secret from the other parties.
- Even if some adversary has power to corrupt some subset of the parties

The Millionaire's Example



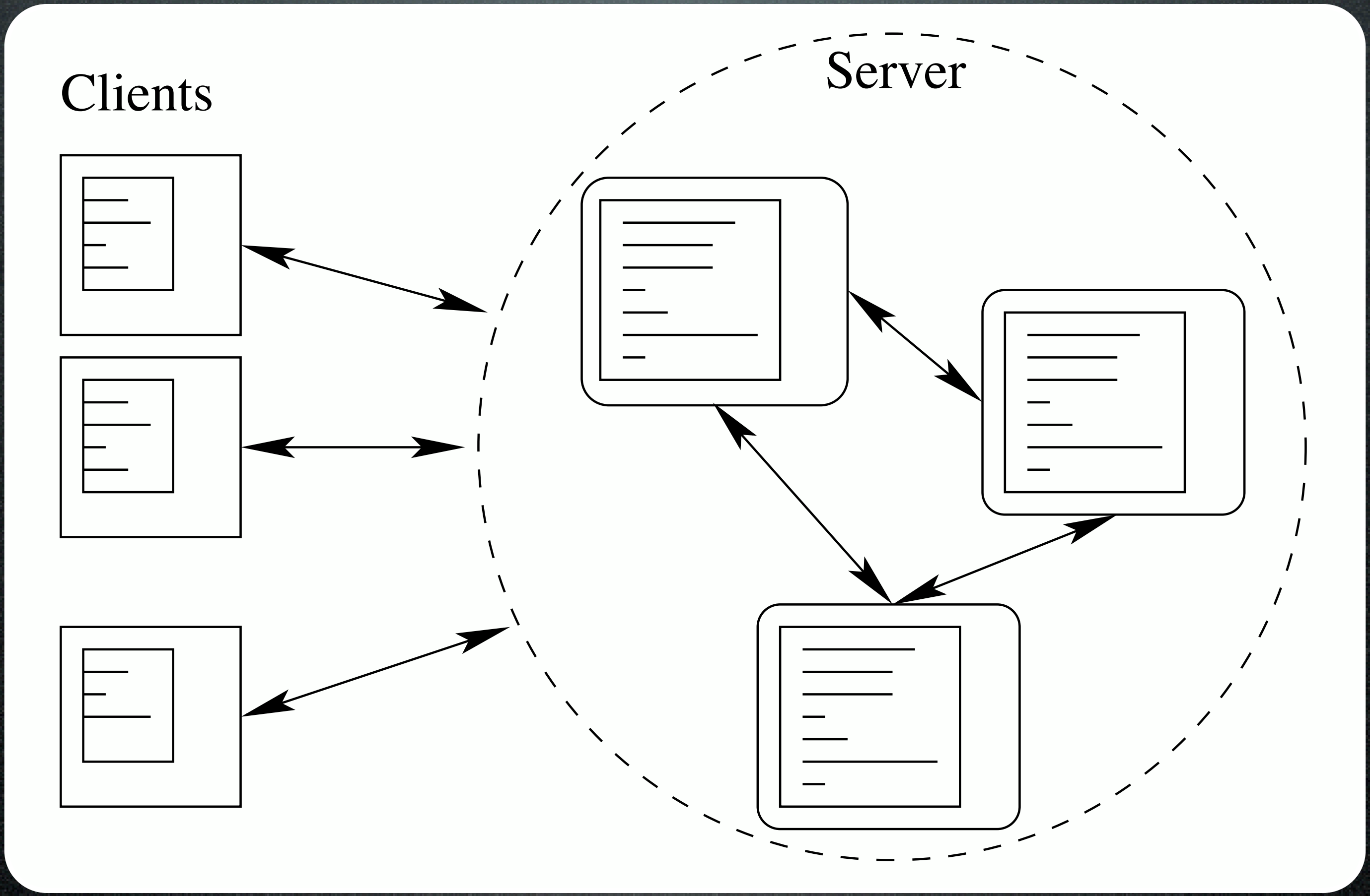
SMC Solves Problems

- Auctions
- Distributed Voting
- Matchmaking
- Benchmarking

Overview

- Secure Multiparty Computation
- **SMCL Concepts**
- An example
- Security - what, why
- Efficiency
- Future Work
- Conclusion

Conceptual Model



Values

Clients:

Private values

- Booleans
- Integers
- Records

Server:

Public & Secret values

- Booleans
- Secret booleans
- Integers
- Secret integers
- Records
- Client identity
- Secret client identity

Communication

Clients:

Tunnels:

- Asynchronous
- put and get functions
- Primitive types only
- Data encrypted
- Secret data - shared and encrypted

Functions:

- Synchronous
- Primitive types only
- Invoked by server

Server:

Tunnels:

- Accessed via client identity
- put and get functions

Client Identity

Clients:

Server:

Groups of clients:

- A set of clients
- All of the same kind
- Iterated using a for loop
- Uniform treatment of clients
- Secrecy of client identity
- Specified externally

Overview

- Secure Multiparty Computation
- SMCL Concepts
- **An example**
- Security - what, why
- Efficiency
- Future Work
- Conclusion

SMCL

The Millionaire's Example

```
declare client Millionaires:
  tunnel of sint netWorth;

  function void main(int[] args) {
    ask();
  }

  function void ask() {
    netWorth.put(readInt());
  }

  function void tell(bool b) {
    if (b) {
      display("You are the richest!");
    } else {
      display("Make more money!");
    }
  }
```

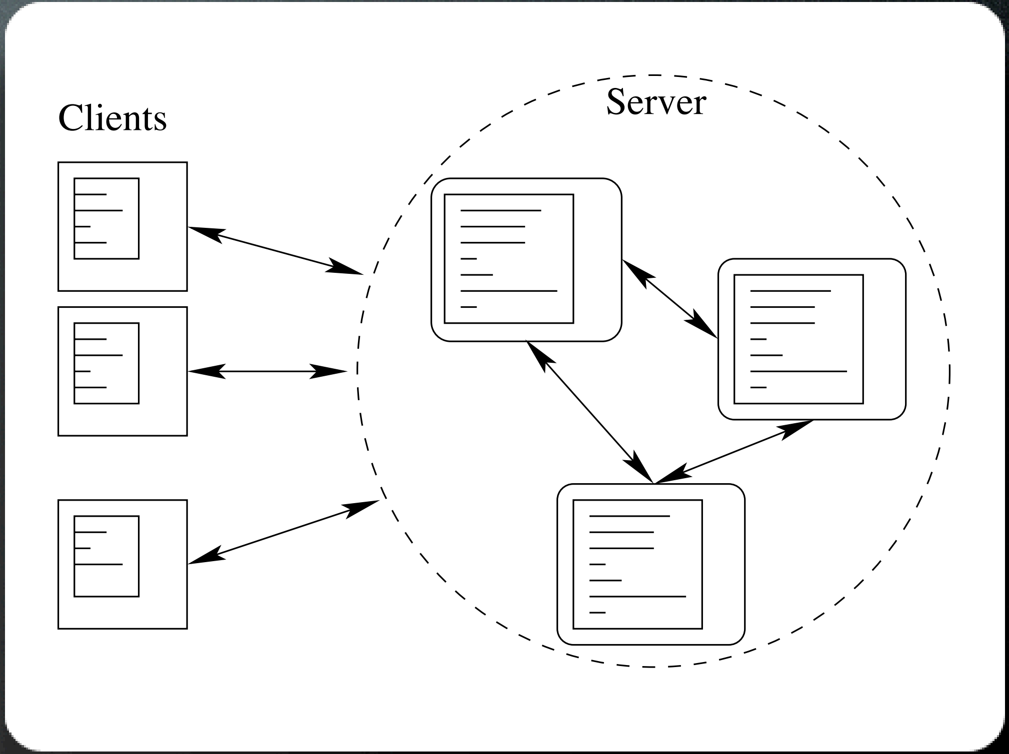
```
declare server Max:
  group of Millionaires mills;

  function void main(int[] args) {
    sint max = 0;
    sclient rich;

    foreach (client c in mills) {
      sint netWorth = c.netWorth.take();
      if (netWorth >= max) {
        max = netWorth;
        rich = c;
      }
    }

    foreach (client c in mills) {
      c.tell(open(c==rich|rich));
    }
  }
```

Overview

- Secure Multiparty Computation
- SMCL Concepts
- An example
- **Security - what, why**
- Efficiency
- Future Work
- Conclusion

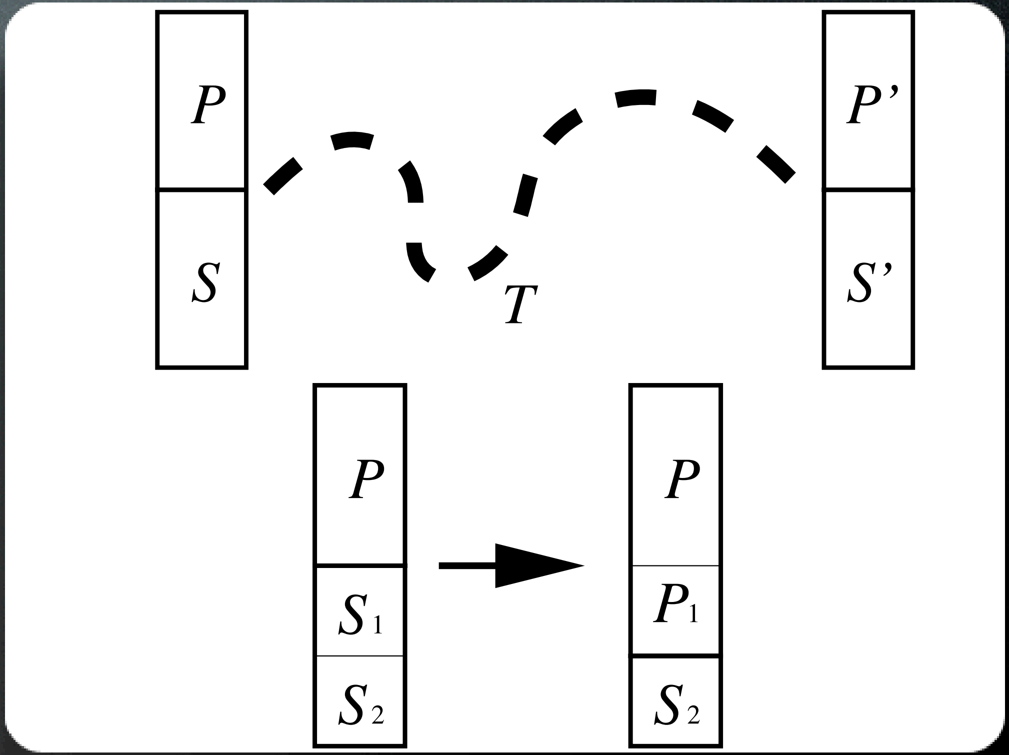
Security

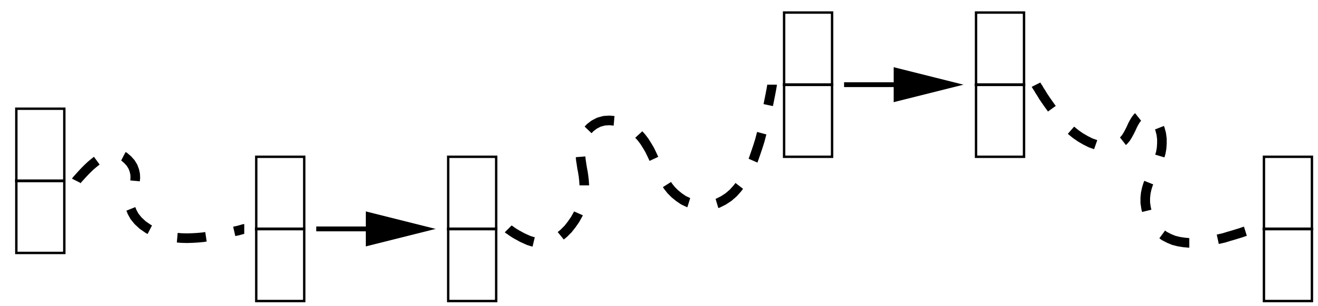
- Identity property
- Commutative property
- Adversary may:
  - Observe physical state of the server
  - Not observe private and secret values



Adversary Traces

- A sequence of states of an entire computation
- Secret values are masked out
- Private state of clients not available
- No declassification

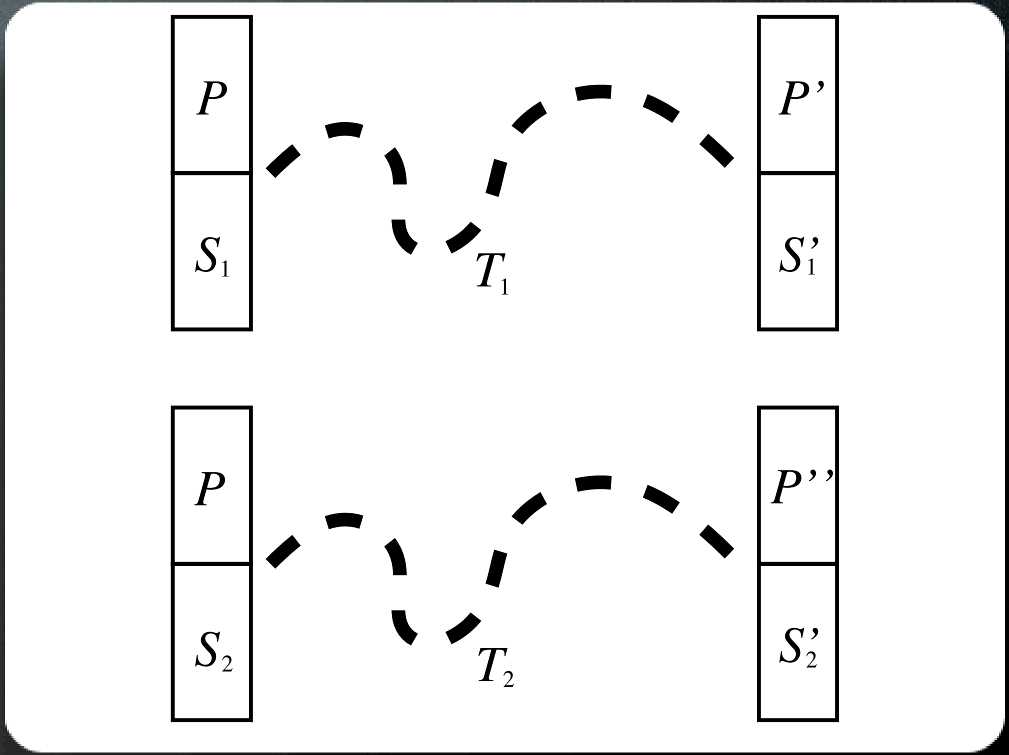




Adversary Traces (cont')

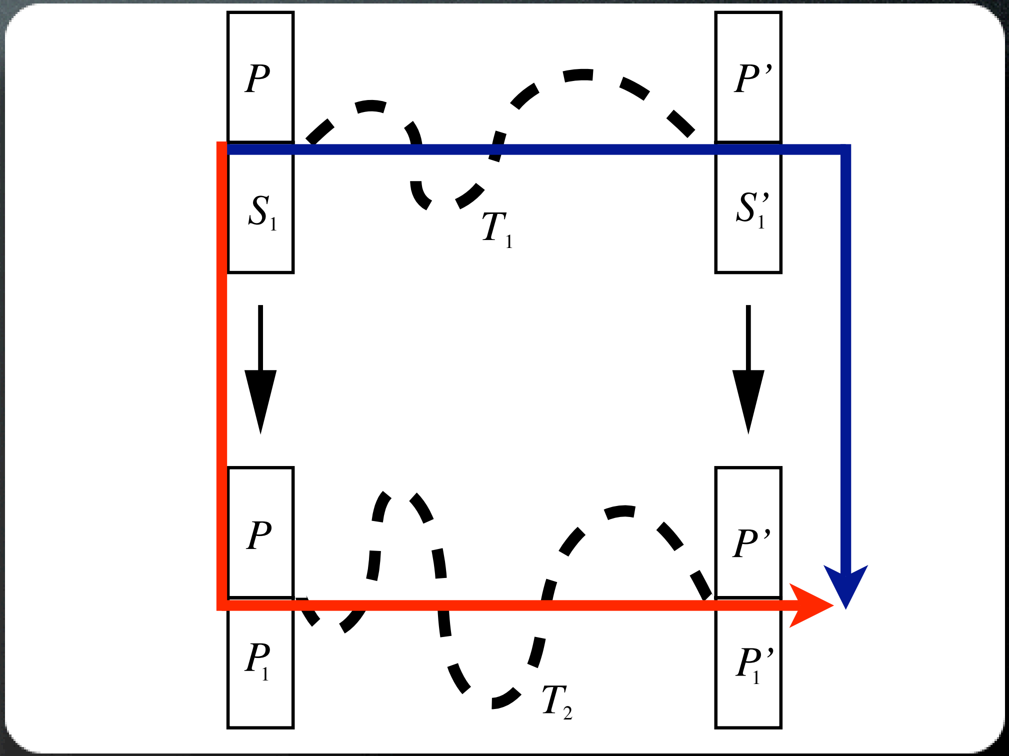
Identity Property

- p’ = p’’ - Low equiv.
- Traces must be identical
- Prevents attacks which are a function of the trace (e.g. timing)
- Requires basic operations independent of arguments



Commutative Property

- Soundness of secret representation



Ensuring Security

- Carefully crafted semantics
- Static analysis of well-typed SMCL programs

Semantics

- Conditionals are a source of differences in trace
  - Execute both branches
  - Termination

```
if (b) {
  x = y;
}
else {
  x = z;
}
```

x = b*y + (1-b)*z

$$\frac{G \vdash \langle C_2, S\rangle \ \rightarrow_{COM_{sv}} \ \langle C_2', S'\rangle}{G \vdash \langle \textbf{if}(\boxed{v})\ \{\}\ \textbf{else}\ \{C_2\}, U_{then}, S\rangle \ \rightarrow_{COM_{sv}} \ \langle \textbf{if}(\boxed{v})\ \{\}\ \textbf{else}\ \{C_2'\}, S', U_{then}, S\rangle}$$

(IF-SBOOL-ELSE)

$$\frac{\begin{array}{c}\sigma.S' = S[x \mapsto \boxed{v} * U_{then}(x) + (1 - \boxed{v}) * U_{else}(x)] \\ \forall x \in S | U_{then}(x) = v \ = U_{else}(x) \vee U_{then}(x) = \boxed{v'} \ \wedge \ U_{else}(x) = \boxed{v''}\end{array}}{G \vdash \langle \textbf{if}(\boxed{v})\ \{\}\ \textbf{else}\ \{\}, U_{else}, U_{then}, S\rangle \ \rightarrow_{COM_{sv}} \ \langle \sigma.S'\rangle}$$

(IF-SBOOL-PHI)

Hoistability

- Branches must agree on public side-effects
  - Assignment to public variables
  - Communication
  - Function calls
- While loops and recursion with secret condition - not allowed

$$\frac{\Gamma_t \vdash e : (\text{bool}, \mathsf{S}, \nu, \iota o)\text{-exp} \qquad \Gamma_t \vdash C_1 : (\mathsf{PL}, \mathsf{NIO})\text{-cmd} \qquad \Gamma_t \vdash C_2 : (\mathsf{PL}, \mathsf{NIO})\text{-cmd}}{\Gamma_t \vdash \textbf{if}\ (e)\ \{C_1\}\ \textbf{else}\ \{C_2\} : (\mathsf{PL} \mathbin{\bar{\wedge}} \nu, \iota o)\text{-cmd}}$$

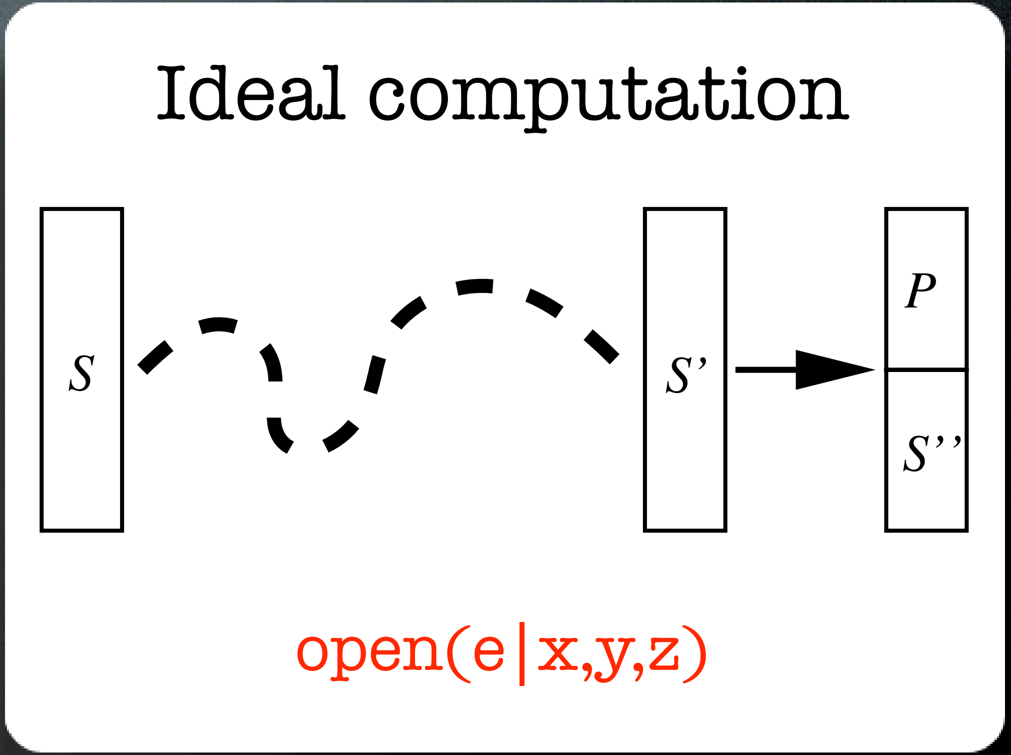
(TIf-secret)

$$\frac{\Gamma_t \vdash e : (\text{bool}, \mathsf{P}, \nu_0, \iota o_0)\text{-exp} \qquad \Gamma_t \vdash C_1 : (\nu_1, \iota o_1)\text{-cmd} \qquad \Gamma_t \vdash C_2 : (\nu_2, \iota o_2)\text{-cmd}}{\Gamma_t \vdash \textbf{if}\ (e)\ \{C_1\}\ \textbf{else}\ \{C_2\} : (\bar{\bigwedge}_{i=0}^{2} \nu_i, \underline{\bigvee}_{i=0}^{2} \iota o_i)\text{-cmd}}$$

(TIf-public)

Semantic Security

- Ideal computations are inefficient
- Prove that a pragmatic version reveals same information as the ideal version
- Assist the programmer



Overview

- Secure Multiparty Computation
- SMCL Concepts
- An example
- Security - what, why
- **Efficiency**
- Future Work
- Conclusion

Efficiency

Ideal

```
sint x = 17;
sint a = 42;
sint b = -5;
sint c = 87;
sint p = a*(x*x) + b*x +c
sint sign = 0;
int output;
if (p<0) sign = -1;
if (p>0) sign = 1;
output = open(sign|p);
```

Pragmatic

```
int x = 17;
sint a = 42;
sint b = -5;
sint c = 87;
int p = open(a*(x*x) + b*x +c|a,b,c)
int sign = 0;
int output;
if (p<0) sign = -1;
if (p>0) sign = 1;
output = sign;
```

(parties, threshold)	ideal	pragmatic	public
(3,1)	12 sec	30 ms	< 1 ms
(5,2)	17 sec	65 ms	< 1 ms
(7,3)	30 sec	132 ms	< 1 ms

Future Work

- SMCL
  - Formalize Adversary traces
  - Dynamic groups
  - Secret compound datatypes
  - More elaborate examples
- SecRas
- SVM, SPL...

Conclusion

- A DSL for SMC
  - High-level abstractions
  - Strong security guaranties
  - Useful in practice

Questions?