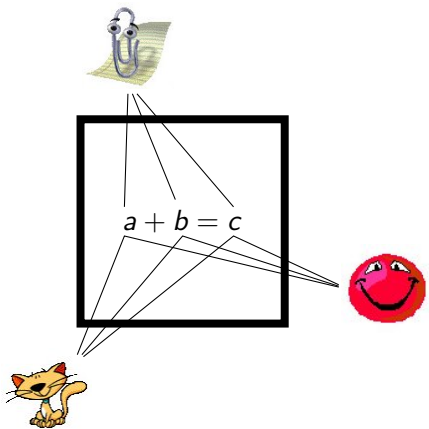
MP-SPDZ at 6

Marcel Keller

CSIRO's Data61

27 November 2024

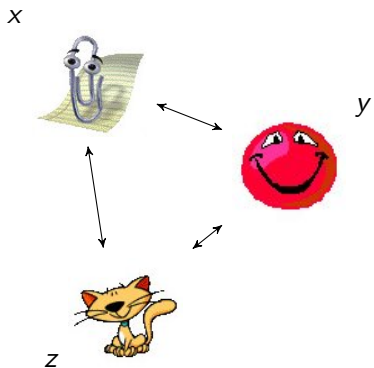
Imagine a Magic Black Box Between a Set of Parties



Parties

- Have handles to values
- Don't know the values
- Can input values
- Can agree on computations creating new values
- Can agree on outputting values

Secure Multiparty Computation: Black Box as Protocol



Wanted: $f(x, y, z)$

- Computation on secret inputs
- Replace black box
- Central questions in MPC
  - How many honest parties?
  - Dishonest parties still follow the protocol?
- MP-SPDZ supports > 40 protocol variants across all properties

MP-SPDZ at CCS'20

- 30+ protocol variants
- Live preprocessing
- Mixed circuits
- C++ API
- Python API
- Dynamic optimization: like HyCC (CCS'18) but all the way
- Basic machine learning

MP-SPDZ since CCS'20

- Complex machine learning (e.g., AlexNet training)
- Decision tree training
- Secure shuffling
- Replace KOS with SoftSpokenOT
- Fantastic Four
- ATLAS protocol (CRYPTO'21 update on DN07)
- Dealer protocol (popular in privacy-preserving ML)
- Distributed key generation for homomorphic encryption
- Simpler interfaces for machine learning (PyTorch integration)
- ARM support
- Bytecode reusability

Compilation with Budget (HyCC)

What to do with loops in MPC?

Consecutive execution slow due to cost of communication rounds

Complete optimization has limits: compilation time, RAM/disk usage

Use a budget!

1. Unroll loop until budget exceeded
2. Optimize unrolled loop
3. Use sequential execution on top of unrolled loop

Example

Need 1000 repetitions: optimize 100, call 10 times

Bytecode Reusability

Mathematical Building Blocks

- Comparison, exponentation, logarithm...
- In C/C++: simple call of function or CPU instruction
- In MPC: want parallelization so no simple function calls

MP-SPDZ

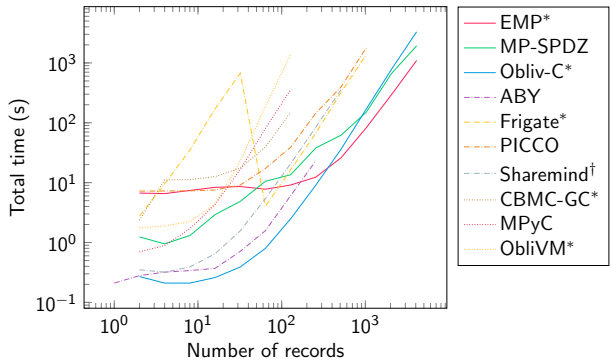
- Find and combine parallelizable invocations of building blocks
- Compile functionality once depending on protocol options
- Reuse bytecode object (faster compilation, lower disk usage)
- Extra benefit: calling high-level code from C++ (new in version 0.4.0)

Crosstabulation Example

```
for i in range(n):
  for j in range(n):
    if x[i] == y[j]:
      sums[cat[i]] += val[j]
```

- Example used by Hastings et al. (S&P'19) to evaluate MPC compilers
- Not necessarily optimal but good test case
  - Nested loops
  - Quadratic run-time

Crosstabulation Benchmarks



Time to compile and run cross-tabulation with different MPC frameworks.
* denotes a garbled circuit implementation, and † denotes an emulation.

Secure Shuffling

Have Secret-shared list $[x_1], \ldots, [x_n]$

Want
- Secret-shared list $[x_{\pi(1)}], \ldots, [x_{\pi(n)}]$
- π random and secret permutation
- Ability to repeat for both π and π^{-1}

Approaches

- Switching networks (Waksman)
- Permute by maximimally unqualified set (honest majority only)
- Dual execution for malicious security (akin SPDZ-wise)
- Recent advances for 2PC using MPC-friendly PRFs (no available implementation)

Application of Secure Shuffling: Sorting

Secret Permutation Operation

Have $[x_1], \ldots, [x_n]$ such that $x_1 = \pi(1), \ldots, x_n = \pi(n)$
$[y_1], \ldots, [y_n]$

Want $[y_\pi(1)], \ldots, [y_\pi(n)]$ without revealing π

1. Reveal secret shuffle ρ of $[x_1], \ldots, [x_n] \Rightarrow$ public description of $\rho \circ \pi$
2. Apply $\rho \circ \pi$ to $[y_1], \ldots, [y_n]$
3. Revert secret shuffle ρ on $[y_{(\rho\circ\pi)(1)}], \ldots, [y_{(\rho\circ\pi)(1)}]$

Radix Sorting

Build permutation needed to sort bit by bit, then apply to data

Application of Secure Shuffling: Decision Tree Training

Problem

- Decision trees are trained level by level where samples are sorted into nodes
- Naive MPC has complexity $O(\#nodes \cdot \#samples)$

Solution by Hamada et al., PETS'22

- Sort samples by node at every level and use secret node markers
- Heavily relies on shuffling

From KOS15 to SoftSpokenOT

KOS15

OT extension with malicious security based on IKNP with simple check
Security claim broken by SoftSpokenOT paper

SoftSpokenOT

- More sophisticated OT extension using codes different to IKNP
- Parameter determining trade-off between computation and communication
- Integrated in MP-SPDZ via libOTe

Reception

SoftSpokenOT considered secure *and* more flexible but some keep talking about KOS. Simplicity?

Outlook

- MP-SPDZ remains popular (citations/GitHub issues)
- Unmatched protocol variety
- Unmatched programmability?