**ObliVM: A Programming Framework for Secure Computation**

**Secure Computation**
- Two parties jointly perform computation over their sensitive data.
- No party learns anything about other’s sensitive data other than output.
- More than three decades old problem [1]

**Challenges**
Hard to program: secure computation protocols require to represent the program in circuits.

Low efficiency: automated compiler (e.g., I2D) to convert a normal C-like program into circuits will incur a big overhead for random memory accesses.

Not scalable: efficient secure computation protocols usually require experts cryptographers to spend a long time to develop.

**RAM-model Secure Computation**
- A naïve approach to convert a random memory access is to perform a linear scan, and thus incurs an overhead linear to the memory capacity. *Inefficient!*
- Existing work (e.g., [3]) relies on Oblivious RAM (ORAM) to automate RAM-model secure computation, which reducing the overhead to poly-logarithmic. It also identifies opportunities to avoid ORAM, in a memory trace oblivious program [4], to further reduce the overhead to be constant.

**ObliVM:**
- Non-crypto experts can use `obliVM-lang` to develop apps.
- `obliVM-lang` is a source language, and can be compiled into optimized libraries.
- `obliVM-SC` is the backend system which performs oblivious data structures.

**ObliVM Approach**
- Programming Language Features
  - Programming Language Features
  - Efficient Implementation
  - New Theoretical Results

**ObliVM Architecture**
- `obliVM-lang` source language
- Compiler
- Non-crypto expert application developer
- Compiler
- Infrastructure/library developer
- Impl. optimized gadgets: Impl new SC protocols
- Impl. optimized libraries: Define new naive functions
- ObliVM-SC Backend

**ObliVM Today**
- Same Tasks
- Ridge Regression
- 5 researchers 3 weeks
- 1 graduate student-day
- 10x20x better performance

**Evaluation**
- Achieve up to $10^6 \times$ speedup with respect to the best automated approach
- As low as 3.5x overhead on top of non-secure baseline

**How far from practical?**
- Distributed GWAS: $130 \times$
- Binary Search on 1GB Database: Today 7.3 secs/query
- Future: 0.3 secs/query

**Adoption**
- Privacy-preserving data mining and recommendation system
- Computational biology, privacy-preserving microbiome analysis
- Privacy-preserving Software-Defined Networking

**References**

---

**Additional Information**
- Maryland
- USC
- Universidad de Chile
- Technicolor
- IDASH
- Communication Sciences Institute
- Ciphering MIPS processor