1. Context & Motivation

In cloud computing, users submit private data to be computed upon by unvetted cloud software.

Problem: Software causes data-dependent behaviors which are visible outside the chip (e.g., I/O requests, power consumption, execution time).

2. Big Ideas

1.) Obfuscated Program Execution

Design processors to generate dummy work, to hide what programs are actually doing (i.e., make every program "look" the same).

2.) Public Knob-based Processor Design

Problem: making all programs "look the same" incurs large overheads.

Solution: design processors to emit signals that are purely a function of a set of user/attacker tunable knobs.

3. Power Channel

- Gate Level Obfuscation
  - Inputs: Public untrusted program, Initial Knob settings $K_1, K_2$.
  - Dynamic knob choices $K_d$ (if $d > 0$).
  - Bit-Leakage Limit $L_d$.
  - Encrypted Inputs (i.e., $\#\$SNF\#$).

- Cache Protocol Level Obfuscation
  - Exploit blocking behavior w/ Knobs

- Hardware Oblivious RAM (ORAM) with Bounded Timing Channel Leakage:
  - Chip boundary
  - Candidate rates
  - Leakage limit ($L_d$)

4. I/O Channel

- Outputs: Encrypted outputs (i.e., $\#\#\#\#\#\#\$)

5. Results

- Core model: In order, single issue.
- On-chip Memory:
  - 1 cycle hit, 32 KB L1 I/D Caches (4 way).
  - 10 cycle hit, 1 MB shared/unified L2 (16 way).
- ORAM latency:
  - 1488 cycle hit/64 Byte cache line.

- [Power+IO obfuscation, static knobs] Pareto curves when changing (L1,L2,ORAM) knobs.
- [Power+IO obfuscation, static knobs] lbq benchmark, varying knob policy.

- [IO obfuscation only, dynamic knobs] base_oram has no timing protection; static_500 (static knobs) = 500 cycles between each access.