Outsourcing Private RAM Computation

Craig Gentry Shai Halevi
Mariana Raykova Daniel Wichs

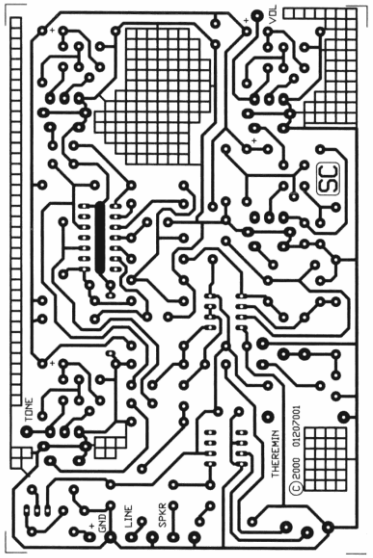
Private Outsourcing

- Client wants to leverage resources of a powerful server to compute $f(x)$ without revealing x.

- Efficiency Requirements:
  - Client works much less than computing $f(x)$
  - Server does about as much work as computing $f(x)$

Private Outsourcing

- Private outsourcing is possible using FHE...
- But FHE works over *circuits* rather than *RAM programs*.



Private Outsourcing

- Private outsourcing is possible using FHE…

- But FHE works over *circuits* rather than *RAM programs*.
  - RAM complexity << circuit complexity (T vs. T^2)
  - For programs where "data resides in memory", the gap can be fully exponential (e.g., Google search).

- Note: using ORAM, can run computation on outsourced data where client & server work as hard as the RAM.

Our Work

- First constructions that allow private outsourcing of RAM computation.
  - Client work $\approx$ input size $|x|$.
  - Server work $\approx$ RAM run time of $f(x)$.

Our Work

- "basic" construction from *iO*
  - Client does one-time preprocessing for a program, then can outsource many independent computations for cheap.
- "best case" construction from a variant of *diO*.
  - Client can also outsource a large database.
    Each computation can read/write to the database.
  - No pre-processing for the program.

“Reusable Garbled RAM”

- Program P → Garbled $\tilde{P}$
  - Client “preprocessing” can be related to RAM run-time of P.
- Input x → Garbled $\tilde{x}$
  - Client “online work” related only to $|x|$
- Garbled $\tilde{P}$+$\tilde{x}$ → $P(x)$ and nothing more
  - Server work related to RAM run-time of P.
- Prior Work: “one-time” garbled RAM. [LO13,GHLORW14]
  - One garbled input per garbled program. Not useful for outsourcing.
- New: “reusable” garbled RAM.
  - Many garbled inputs for the same garbled program.

Our Approach

- Combination of:
  - "One-time Garbled RAM" [LO13,GHLORW'14]
  - "Reusable garbled circuits" [GKPVZ'13]

- Idea: Create a reusable garbled circuit that gets x computes a fresh one-time garbled RAM: $\tilde{P}, \tilde{x}$

Main Difficulty

Need to garble circuit with small input, huge output

Want to have small garbled inputs.

- Not achieved by known constructions [GKPVZ13].
- Show: not possible with simulation-based security.

- New: make due with weaker notions of security for garbled circuits: "distributional indistinguishability"

- New: constructions of such reusable garbled circuits with "right efficiency" based on obfuscation.
  - Open Problem: weaker assumptions!

Thank You!